UNIVERZITA PALACKÉHO V OLOMOUCI
Pedagogická fakulta
Katedra technické a informační výchovy

BAKALÁŘSKÁ PRÁCE

Multimediální aplikace pro výuku intonace

2015

Eva Strnadová

Vedoucí práce: Mgr. Jan Kubrický, Ph.D.

Studijní obor: Informační výchova se zaměřením na vzdělávání, Hudební kultura se zaměřením na vzdělávání

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a použila jen uvedené bibliografické a elektronické zdroje.

V Olomouci 22. dubna 2016 .....

podpis

Ráda bych poděkovala Mgr. Jana Kubrickému, Ph.D. za vstřícnost a trpělivost při vedení bakalářské práce. Mé poděkování patří také Ondřeji Crhovi za odborné rady ohledně praktického výstupu práce.

Obsah

Úvod

Bakalářskou práci věnuji multidisciplinárnímu tématu výuky intonace a sluchové analýzy s podporou moderních technologií, které je mi jako studentce informační výchovy a hudební kultury blízké.

Na myšlenku realizace hudební aplikace mne přivedly vlastní zkušenosti s intonačním výcvikem v prvních dvou ročnících vysokoškolského studia. Domácí příprava studentů zde silně naráží na omezené možnosti kontroly správnosti takových cvičení, protože to vyžaduje zvukovou kontrolu, realizovanou většinou na hudebním nástroji. Nelze nadále popírat fakt, že přístup k počítači nebo chytrému mobilnímu zařízení má mnohonásobně více studentů než k naladěnému klavíru.

Intonační software a aplikace zůstávaní tématem, o kterém bylo v posledních 15 letech mnoho řečeno, ale málo skutečně vykonáno. Jako hlavní cíl mé bakalářské práce jsem vytyčila tvorbu aplikace pro podporu intonačního a rytmického výcviku, která může sloužit jako doplněk studia tonální písňové metody Ladislava Daniela. Jednotlivé kapitoly bakalářské práce se zabývají procesem vývoje vytvářené webové aplikace od analýzy vstupní situace přes návrh po samotnou implementaci konkrétního řešení.

První kapitola *Intonační a rytmický výcvik* uvádí do hudební stránky problematiky, jejím cílem je objasnit podstatu hudební intonace na takové úrovni, která umožní pochopení základních prvků a principů vytvářené aplikace. Následuje *Analýza požadavků*, která nejprve nastiňuje obecnější problémy, se kterými se potýká software a online produkty z oblasti intonační výchovy a z nich vyvozuje konkrétní požadavky na vyvářenou aplikaci.

Vybrané nástroje pro tvorbu aplikace ve stručnosti uvádí třetí kapitola *Použité externí komponenty*. Zde kladu zvláštní důraz na notační API VexFlow, protože nejde o běžně známou knihovnu a zvlášť v českém jazyce narážíme na nedostatek informací. Kapitola *Práce s VexFlow* přináší první celistvý český text pojednávající a tvorbě notace s tímto API. Poslední kapitola *Návrh a implementace aplikace* mapuje samotný tvůrčí a programátorský proces, provázána s komentovanou dokumentací zásadních částí v příloze práce.

1 Intonační a rytmický výcvik

Zpěv z not (tj. intonační a rytmický výcvik) je nejen cílem, ale především jedním z hlavních prostředků rozvoje hudebních schopností. (Daniel, 2010, s. 39) K intonaci se vztahuje velké množství metod, které můžeme rozdělit na tonální a intervalové.

V rámci tonálních metod chápeme tóny jako stupně určité tóniny. Tyto metody považujeme pro začátečníky za vhodnější, protože na rozdíl od intervalových nevyžadují znalost hudební teorie, vedou k čisté intonaci a zpívající snadněji udrží tóninu. (Havránková, 2013, s. 9) Za u nás nejpoužívanější uvádí Havránková (2013, s. 12) číselnou metodu Čestmíra Staška, nápěvkovou metodu Františka Lýska a tonální písňovou metodu Ladislava Daniela. Poslední jmenované bude věnována webová aplikace v této bakalářské práci.

1.1 Tonální písňová metoda Ladislava Daniela

Koncept tonální písňové metody vychází z bazální skutečnosti, že i nehudebník dokáže intonovat a poznat stupeň tóniny na základě znalosti písně na tento stupeň začínající. Kupříkladu žák bez hudební průpravy, kterému udáme tóninu C dur a požádáme jej o zazpívání tónu *g* nebo pátého stupně, si nebude vědět rady. Pokud by však měl zazpívat písničku *To je zlaté posvícení*, sám přirozeně od *g* začne. (Daniel, 1994, s. 3) Osobně se domnívám, že jde o zdaleka nejpřístupnější z uvedených metod, vhodnou i pro naprosté začátečníky v hudbě obecně. V praxi doporučuje Daniel (2010, s. 39) tuto metodu úzce propojit s rytmickým výcvikem.

Rozšířenost tonální písňové metody (a tedy i počet potenciálních uživatelů vytvářené aplikace) je větší, něž by se mohlo na první pohled zdát. Jak uvádí Gazdová (1998), díky Danielovým pedagogickým aktivitám se stala obecně známou především mezi českými a slovenskými pedagogy, ale její ozvuky můžeme pozorovat v celé střední Evropě a značně také ovlivnila pojetí hudební výchovy v Dánsku.

Metoda čerpá ze základních principů přirozené lidské hudebnosti. Podle Daniela (2010, s. 39) stačí ke zvládnutí intonace dvě základní dovednosti:

1. Umět sluchem poznat a zazpívat každý tón dané tóniny.
2. Umět zazpívat a zapsat diatonické postupy.

Za účelem dosažení těchto dovedností pěstujeme v žácích dva principy vybavování představ tónů. V první řadě jsou to tzv. *volné nástupy*, kterými rozumíme

první tón melodie a každý další tón po větším intervalu. Pro vybavení slouží *opěrné písně*, tedy obecně známé písně, u kterých žák ví, na který začínají stupeň. Diatonické postupy jsou potom vzestupné a sestupné postupy a opakované tóny, které následují po volném nástupu. Pro jejich vybavení není třeba žádná zvláštní pomůcka, protože jejich správnou interpretaci přirozeně zajistí názornost hudebního písma a tonální cítění intonujícího. (Daniel, 2003, s. 230)

Základním požadavkem na tato cvičení tedy je, aby obsahovala pouze diatonické postupy (tj. vzestup nebo sestup po stupnici) a volné nástupy odpovídající studijní úrovni žáka.

S současné době je pro nácvik této metody dostupných několik pomůcek z dílny Ladislava Daniela, přičemž všechny vyžadují následnou kontrolu úspěšného zvládnutí cvičení pomocí hudebního nástroje.[1] Jde například o tzv. melodické kostky, různé varianty učebnic a cvičebnic (ty však poskytnou pouze omezený počet cvičení). Za zmínku stojí zejména velmi nápaditá Intonační cvičení I. a II.,[2] jde o sešity hudebních úryvků ve spirálové vazbě připojené shora tak, aby úryvky šly vertikálně rozstříhat a libovolně kombinovat. Tato cvičení do značné míry inspirovala aplikaci vytvářenou v této bakalářské práci.

1.2 Softwarová aplikace písňové metody

Autor této intonační metody, profesor Ladislav Daniel, sám vytvořil návrh programu ke zlepšení výuky intonace, který v roce 2003 publikoval v odborném občasníku e-Pedagogium.[3] K realizaci jeho návrhu zřejmě nikdy nedošlo.[4] Dalším impulzem byly diplomové práce Jiřího Hokeše *Návrh software pro výuku intonace a sluchové výchovy* (2005) a na něj navazujícího Jana Křupy *Nové podněty k vytváření počítačových programů pro výuku intonace a sluchové výchovy* (2008), ani z těch ovšem nevzešel žádný praktický výstup.

Jako možný důvod vidím silně hudebně-metodologický náhled autorů bez přihlédnutí k náročnosti vývoje dané aplikace. Všichni například svůj návrh staví

[1] Jako alternativní možnost se nabízí využití moderních technologií, například mobilní aplikace simulující klaviaturu.
[2] Daniel, Ladislav. *Intonační cvičení I.* Praha: Schott Music Panton, 2009.
Daniel, Ladislav. *Intonační cvičení II.* Praha: Schott Music Panton, 2011.
[3] Daniel, Ladislav. 2003. Výuka zpěvu z not na počítači. *e-Pedagogium.* 3. roč., 2003, č. 1 - mimořádné, stránky 230-232.
[4] Přestože Křupa (2008, s. 10) na něj odkazuje jako na „realizovaný" projekt, k 20. 3. 2016 jsem v dostupných pramenech nedohledala žádnou zmínku, která by to potvrzovala.

na analýze zvukového signálu pro vyhodnocení intonační správnosti zpěvu. Problémy její praktické realizace okomentoval doc. PhDr. Vladimír Rambousek, CSc., vedoucí Katedry informačních technologií a technické výchovy UK v Praze, takto:

„*Obvykle se k tomuto používá Fourierova analýza, kdy je vyhledána „nosná frekvence", tedy ta, která tvoří hlavní část zvuku. Bohužel hlas (na rozdíl například od zvuku houslí, kytary apod.) se skládá z velkého množství frekvencí a jeho analýza není triviální. Pokud má tedy program analyzovat hlasový projev z hlediska správné intonace, je vytvoření správně fungujícího programu poměrně složité. Lze doporučit spíše nalezení existujícího řešení na některém z pracovišť, které se zabývají analýzou zvuku/hudby, např. nějaké pracoviště na Keplerově univerzitě v Linci.*“ (Rambousek In: Křupa, 2008, s. 52)

1.2.1 Podněty Ladislava Daniela

Podle Daniela by hlavním principem a obsahem intonační aplikace měla být „*metodicky progresivně uspořádaná řada sluchových a intonačních cvičení.*“ (2003, s. 230) Obtížnost, tempo postupu a typ cvičení si volí sám žák, počítač kontroluje správnost zápisu nebo zpěvu a okamžitě žákovi poskytuje zpětnou vazbu. Čerpání příkladů navrhuje z výše zmíněných *Intonačních cvičení I. a II.* Daniel (2003, s. 232) uvádí, že několik let metodicky zpracovával obsah programu, který obsahuje celé učivo intonace (např. i modální tóniny), tyto materiály bohužel zůstaly po jeho smrti v roce 2015 nepublikovány.

1.2.2 Podněty Jiřího Hokeše a Jana Křupy

Hokeš (2005) zaměřuje navrhovaný program na začátečníky. Zařazuje náhodně vygenerované melodie a sluchová cvičení vycházející z vokální intonace. Vše opět řadí do několika úrovní obtížnosti, aby měl žák možnost postupovat vlastním tempem. Křupa (2008) označuje za hlavní přínos Hokešovy práce srozumitelnou metodologickou část. Sám se zaměřuje na nácvik vícehlasu. Pracuje s poměrně inovativní myšlenkou, že žák do mikrofonu zpívá jeden hlas a ostatní mu doplňuje počítač. Ve své diplomové práci navrhuje systém obtížností postupující od izolovaných akordů, přes kánony a cvičení polyfonního charakteru až po čtyřhlasou homofonii.

2 Analýza požadavků

2.1 Obecné požadavky a podobné aplikace na trhu

Oblast technické podpory výuky a učení prošla v posledních dekádách prudkým rozvojem. I v poměrně rezistentní oblasti hudební výchovy se od druhé poloviny 90. let začaly pomalu objevovat aplikace cílené na rozvoj hudebnosti.[5] Aplikace pro výuku intonace nebo sluchové analýzy cílí na poměrně malou uživatelskou skupinu, jejich komerční vývoj tak není v lokálním českém prostředí pro firmy atraktivní. K dispozici je ale velký počet produktů v anglickém jazyce různorodých kvalit a obsahové šíře.

V této kapitole se zaměřuji na obecné požadavky kladené na aplikace tohoto typu s přihlédnutím ke specifickým potřebám českého uživatele, ty nejznámější ve stručnosti představuje a hodnotí kapitola 2.1.7.

2.1.1 Přívětivost uživatelského rozhraní

Že základním předpokladem uživatelské spokojenosti, je jednoduché a intuitivní ovládání, platí v souvislosti s jakýmkoli druhem aplikace. Uživatel by se měl v prostředí snadno orientovat a být schopen rychle provést požadovanou akci. (Řezáč, 2014, s. 162-168)

Za jeden z hlavních prostředků úspěšného vývoje přívětivé aplikace považuje Wasielewski (2003 cit. dle Loukotová, 2009, s. 17) porozumění uživateli. Pouze to umožňuje navrhnout aplikaci, která pomůže uživateli dosáhnout jeho přání a cílů. Prostředí mělo být pro uživatele příjemné a bezpečné, zpracovávat rychle a elegantně uživatelské požadavky.

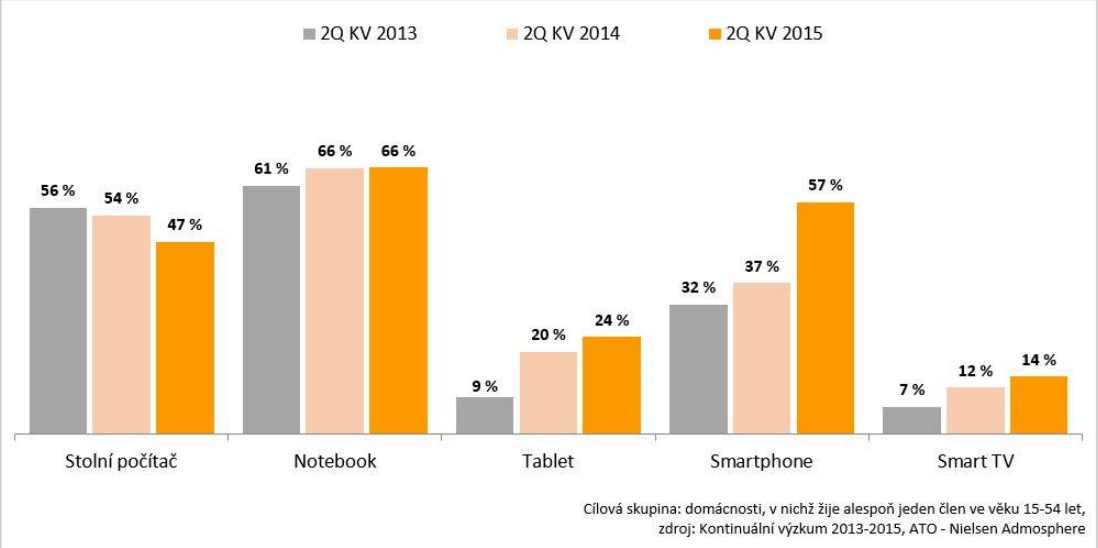
2.1.2 Šíře podporovaných platforem

Jak naznačují evropské (Ericsson, 2015) i české (Nielsen Admosphere, 2015) výzkumy, struktura vlastnictví elektronických zařízení v domácnostech se poměrně rychle mění, a to ve smyslu diferenciace typů zařízení a zvyšování počtu uživatelů, kteří si jejich vlastnictví mohou dovolit. Dalším aspektem je možnost použití aplikace

[5] Například první edice známého intonačního programu EarMaster vyšla v roce 1996.

přímo ve školním prostředí, kde tentýž trend potvrzuje například vlna zájmu o metodu BYOD (Bring Your Own Device) nebo projekty zavádění tabletů do škol.

Z hlediska cílení výukových aplikací tedy považuji za důležité dbát na všechny skupiny uživatelů a umožnit jim použití jejich typu zařízení, multiplatformita se zde jeví jako optimální přístup.



Obrázek 1 - Vybavenost českých domácností elektronickými zařízeními v letech 2013 – 2015 (Nielsen Admosphere, 2015)

2.1.3 Variabilnost výuky

Jak bylo nastíněno v kapitole č. 1, reálná výuka dovednosti zvané „*intonace*“ se zpravidla skládá ze dvou složek – intonace jako takové (tedy schopnosti zpěvu podle notového zápisu) a sluchové analýzy (tedy schopnosti zapsat notový zápis na základě poslechu), úzce ji také provází rytmický výcvik.

V ideálním případě by aplikace měla pojmout všechny druhy žáků od začátečníků po velmi pokročilé a zároveň nabídnout různé typy cvičení. S obtížností samotného cvičení souvisí i požadavek diferenciace cvičení na základě uživatelského nastavení. Jak zdůrazňuje Hokeš (2008, s. 20), v dostupných programech převládají jednodušší cvičení postačující pro rozvoj pouze začátečníkům.

2.1.4 Dostupnost české jazykové lokace

2.1.4.1 Uživatelské prostředí

Jak uvádím v analýze uživatelů (kapitola 2.2), systematickému studiu intonace se v České republice věnují zejména žáci středních a studenti vysokých škol. Ti by měli v cizím jazyku disponovat dostatečnými receptivními dovednostmi, aby dokázali bez větších potíží pracovat v cizojazyčném uživatelském prostředí. Přístupnost v mateřském jazyce zůstává pro studenta výhodou, ale z hlediska použitelnosti není zásadním kritériem.

2.1.4.2 Hudební názvosloví

Intonační software vždy pracuje v určité své části s hudebním názvoslovím. Zde považuji za zásadní dostupnost české lokace, protože dominující tzv. anglické názvosloví se odlišuje ve všech úrovních, a to včetně označení tónů (tón *h* se v angličtině označuje jako *B*, odlišně se také tvoří názvy snížených a zvýšených tónů).[6]

Tímto požadavkem narážíme na jeden z hlavních problémů využitelnosti dostupných intonačních programů, které nejsou přizpůsobeny pro silně minoritní hudební značení, používané pouze v ČR a okolních státech. (Fíbek, 2015)

2.1.5 Návaznost na teoretickou a metodologickou podporu

Podstatou tohoto bodu je požadavek, aby aplikace nebyla pouhým souborem mnoha nahodilých cvičení bez kontextu, ale pracovala pod uceleným metodologickým přístupem. Jedná se jednak o teoretickou podporu obsaženou přímo v daném prostředí, jednak o celkovou svázanost s osvědčenou hudebně výchovnou metodologií.

Domnívám se, že většina potenciálních uživatelů se hudební intonaci nějakým způsobem věnuje i mimo prostředí aplikace. Přichází tedy ovlivněni a případně i odborně školeni v některé z v České republice nejpoužívanějších intonačních metod, které zmiňuje kapitola č. 1. Od aplikace potom očekávají prostředí vhodné k procvičování a rozvoji ve studované metodě, to jim pochopitelně zahraniční produkty nemohou nabídnout.

[6] Podrobnosti k rozdílům v českém, italském a německém hudebním názvosloví např. v Zenkl, 2003, s. 178–180.

2.1.6 Cenová dostupnost

Cenová dostupnost je pro většinu uživatelů významným faktorem při volbě konkrétní aplikace. Vzhledem k tomu, že cílovou skupinou jsou především žáci a studenti, je tento požadavek ještě markantnější. V ideálním případě by taková aplikace měla být (v dostačující kvalitě) zcela zdarma, což však logicky nekoresponduje se systémem trhu.

2.1.7 Nejznámější intonační aplikace

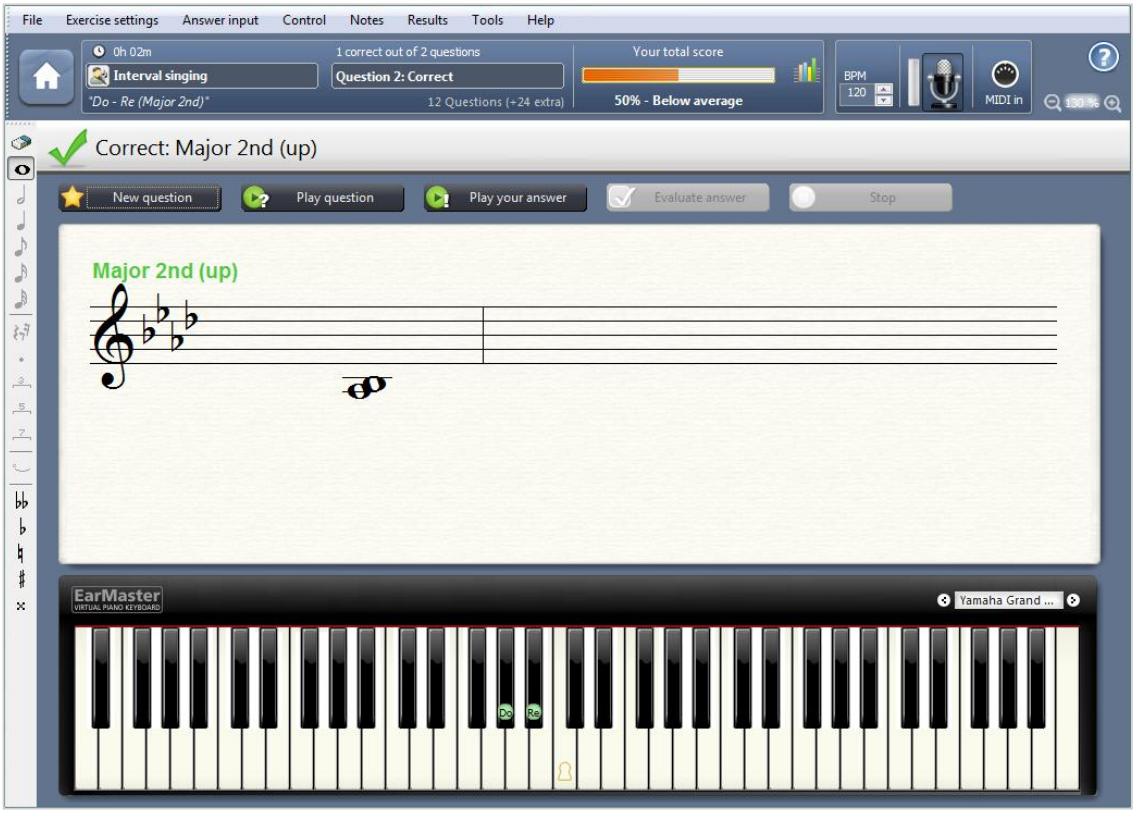
Porovnání aplikací pro podporu hudební výchovy se ve svých diplomových pracích v minulosti věnovali například již zmiňovaní Hokeš (2005) a Křupa (2008). Velmi propracované hodnocení nedávno poskytl Michal Fíbek ve své práci *Interaktivní e-learningová aplikace pro podporu hudební nauky na ZŠ* (2015). V tomto souhrnu vycházím z vlastní zkušenosti a závěrů posledních dvou jmenovaných.

2.1.7.1 EarMaster

Program *EarMaster* (aktuálně ve verzi 6.2) je nabízen v několika edicích, které jsou určeny pro platformy Windows a Mac OS, testování beta verze pro iPad s současnosti probíhá. Aplikace se zaměřuje spíše na individuální výuku,[7] jejím velkým kladem je využití vstupu z mikrofonu k vyhodnocování zpěvu tónů a vytleskávaného rytmu. Zatímco kontrola rytmu funguje spolehlivě, o snímání zpěvu Fíbek (2015) uvádí, že reaguje velmi citlivě v závislosti na kvalitě mikrofonu a hlasu uživatele. Nabízí čtyři varianty kvalitně vypracovaných testů, pojetím nijak nevybočujících mezi nabídkou dalších programů.

Přestože podporuje několik jazyků, čeština mezi ně v nejnovější verzi 6 nepatří,[8] nenabízí ani uspokojivé řešení pro lokaci hudebního názvosloví. Časově neomezená (krabicová) verze se pohybuje v cenové hladině 1600 – 2700 Kč v závislosti na zvolené variantě, v nabídce jsou také cloudové multilicence pro školy.

[7] Verze EarMaster Pro na rozdíl od verze Essential rozlišuje učitelský účet. Ten umožňuje vytvářet vlastní lekce, ale chybí prostředky k přímému ovlivnění výuky. (Fíbek, 2015 s. 21–22)
[8] Fíbek (2015, s. 22) ve své práci zmiňuje předpokládané rozšíření o českou lokaci (a notaci), k datu 18. 4. 2016 ale stále nebyla dostupná.



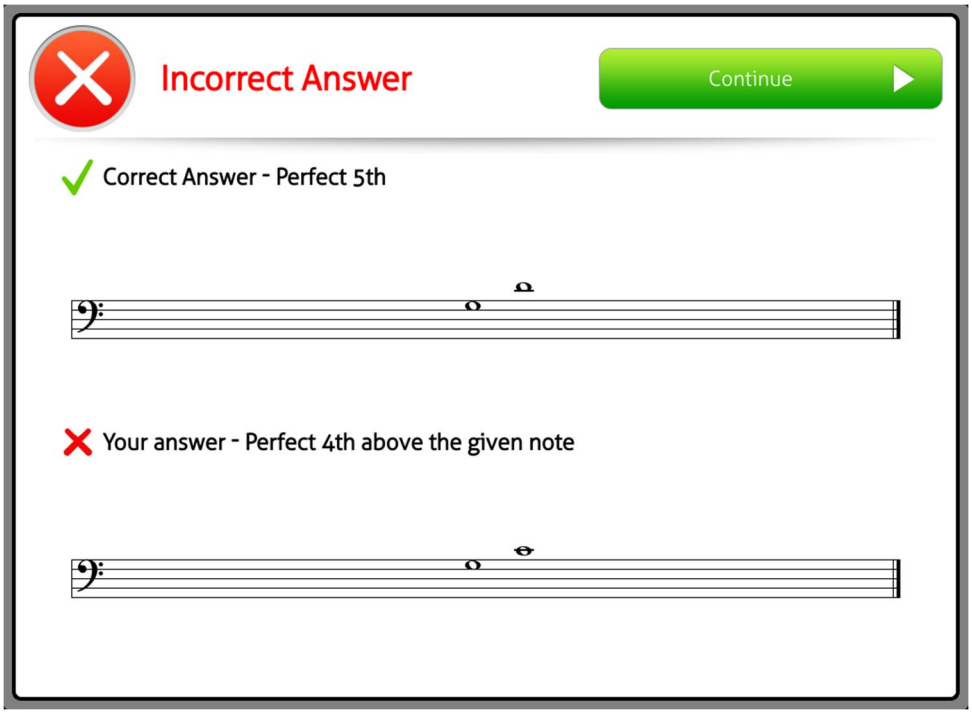
Obrázek 2 – EarMaster 6, modul zpěvu intervalů (screenshot)

2.1.7.2 Auralia & Musition 5

Auralia a *Musition* původně vznikly jako dva samostatné programy společnosti Sibelius, která je spojena se známým notačním programem téhož jména. Od verze 4 jsou prodávány výhradně ve dvoubalení, graficky i pojetím sjednoceny, takže je uvádím jako jeden celek. Cílí na platformy Windows a Mac OS, jednotlivé lekce jsou samostatně k zakoupení pro přístroje Apple. (Rising Software, 2015)

Jde o vůbec nejkomplexnější aplikační prostředí cílené na rozvoj hudebnosti. *Musition* slouží pro výuku hudební teorie, dělí se na 8 obsahových modulů pojímajících začátečníky i pokročilé. *Auralia* je určena ke studiu intonace a sluchové analýzy, má 7 obsahových modulů zaměřených na poznávání různých druhů stupnic, akordů, porovnávání výšky tónů, nácvik rytmu, zpěv a identifikaci intervalů. Pracuje s reálnými hudebními úryvky.

V aplikaci lze při prvním spuštění zvolit formát not podle české notace, ale celé její rozhraní a zadání cvičení jsou dostupné pouze v anglickém a německém jazyce. (Fíbek, 2015) Studenti si tuto dvojici programů mohou koupit v přepočtu za 2 350 Kč, běžní uživatelé za 3 775 Kč, opět jsou nabízeny i cloudové multilicence pro školy.



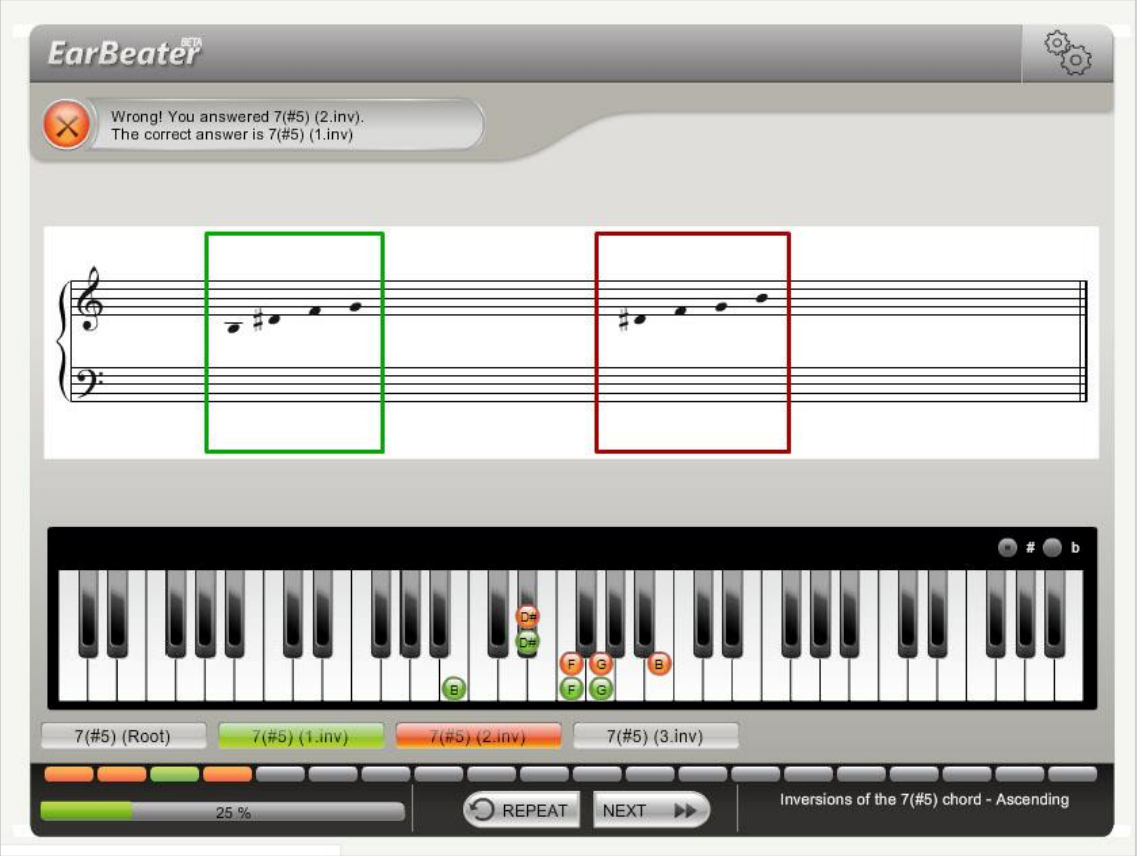
Obrázek 3 – Auralia 5, modul zpěvu intervalů (Rising Software, 2016)

2.1.7.3 EarBeater 1.1.1

Jako jistý protipól předchozím aplikacím uvádím aplikaci *EarBeater*, která obsahem nijak nevyčnívá mezi podobnými aplikacemi, ale svou popularitu mezi uživateli získala především díky online verzi zdarma a množstvím cvičení. Aplikace byla jednorázově vyvinuta v roce 2015, funguje jako neresponzivní online prostředí vhodné pro desktop nebo aplikace pro iPhone a iPad.

Online verze *EarBeater Classic* nevyžaduje při používání přihlášení, nabízí pět modulů zaměřených na sluchovou analýzu, jde konkrétně o porovnávání a poznávání intervalů, akordů, obratů a stupnic. Celkem nabízí 200 jednotlivých lekcí, které jsou svým obsahem pevně dané. Samotná cvičení mají příjemné ovládací prvky a jasnou strukturu.

Podporuje jazyky angličtinu a švédštinu, nikoli však možnost zvolit formát not podle české notace. Aplikace pro iPhone a iPad s některými přidanými funkcemi je ke koupi za v přepočtu 190 korun, online *Ear Beater Classic* zcela zdarma.



Obrázek 4 – EarBeater Classic 1.1.1, modul rozpoznání stupnic (screenshot)

2.2 Analýza uživatelů

Systematická výuka vokální intonace a sluchové analýzy se v českém vzdělávacím systému uplatňuje zejména ve vybraných typech škol sekundárního a terciálního vzdělávání. Jedná se například o konzervatoře, gymnázia s rozšířenou výukou hudební výchovy, střední a vyšší odborné pedagogické školy a vysokoškolské obory hudebně výkonného, hudebně teoretického, hudebně pedagogického a obecně pedagogického zaměření.

U dětí mladších se s ní setkáváme výjimečně, například v souvislosti s hlasovou přípravou v elitních dětských a mládežnických pěveckých sborech. Malou skupinu budou pravděpodobně tvořit i samoukové bez zázemí vzdělávací instituce.

2.2.1 Účel aplikace

Umožnit jednoduchou interaktivní formou nácvik intonace bez nutnosti kontroly vyučujícím nebo hudebním nástrojem.

2.3 Funkční požadavky

Hlavním úkolem aplikace je generovat hudební úryvky vhodné svou strukturou k procvičování tonální intonační metody Ladislava Daniela, která je popsána v předchozí části této práce.

Vytvářená aplikace by měla umět na základě specifikace požadavků od uživatele sestavit intonační cvičení, zobrazit jeho notaci, zvukově navodit tóninu a následně přehrát zvukovou podobu cvičení.

Zároveň by měla uživateli poskytnout dostatečnou teoretickou podporu, zejména při nácviku opěrných písní – to znamená disponovat notovými zápisy vhodných lidových písní s možností navodit tóninu, ověřit první tón a přehrát jejich zvukový záznam.

2.4 Další požadavky

Aplikace by měla:

- mít moderní design a uživatelsky přívětivé rozhraní;
- být co nejméně závislá na platformě z pohledu zařízení a ovládání (dotykové zařízení i desktop ovládaný myší) i prohlížeče;
- stavět na zásadách objektově orientovaného programování;
- být modulární a sestavená s ohledem na plánovaná rozšíření.

3 Použité externí komponenty

3.1 Medoo Framework

Medoo Framework je PHP databázový wrapper distribuovaný pod MIT licencí. Díky své malé velikosti šetří hardwarové prostředky a ze strany klienta zjednodušuje komunikaci s databází tím, že za něj seskládá SQL dotazy. Umí pracovat s různými typy databází včetně MySQL, využívat složitějších dotazů (jako např. spojování tabulek) či používat transakce. (The Medoo Project, 2016) Překvapivě chybí česká podpora nebo aktivní komunita uživatelů.

3.2 Bootstrap

Bootstrap, aktuálně ve verzi 3, je populární frontend framework poskytující nástroje k rychlému a jednotnému vývoji webových projektů. Původně vznikl pro potřeby společnosti Twitter. Jedná se o velmi univerzální nástroj, který vylepšuje vzhled a funkčnost webu úpravou typografie, formulářů, tlačítek, navigace apod. K tomu používá sadu předpřipravených stylů a funkcí, které lze jednoduše aplikovat na HTML kód i bez znalosti CSS a javascriptu. Jedním z nejvýraznější problémů, které Bootstrap pomáhá řešit, je responzivita webu.

Bootstrap je tvořen kaskádovými styly a javascriptem, přičemž CSS lze na webu použít i nezávisle (bez načtení souboru javascriptu). Podporuje použití preprocesoru LESS nebo SASS, které zrychlují načítání, protože jsou v nich obsaženy jen vybrané komponenty, které web skutečně používá. Výhodou zůstává možnost měnit počet těchto komponent v průběhu projektu.

Kladem tohoto frameworku je také velká uživatelská komunita, a to i v České republice, což je příčinou dostatku tutoriálů a diskuzí na témata s ním spojená. Za pozitivum můžeme také označit zdařilou dokumentaci. Na oficiálním webu (Bootstrap, 2016) jsou dostupné rozsáhlé, ale přehledné informace o možnostech aplikace CSS a javascriptu, úpravě zápisu a individualizaci jednotlivých komponent, implementaci kódu apod.

3.3 JS Dynamic Audio Synth

JS Dynamic Audio Synth je nenáročný syntetizátor zvuku s dynamickým průběhem tónu napsaný v javascriptu, který v roce 2013 pod CC licencí publikoval Keith William Horwood.

Na rozdíl od většiny dostupných nástrojů s podobnou funkcí nepoužívá předem nahrané zvukové soubory, ale za použití matematických principů napodobuje plasticitu tónů reálných hudebních nástrojů.[9] Generuje dynamické WAV audio, které převede do schématu *dataURI* a přehraje v prohlížeči s použitím elementu HTML5 audio. (Horwood, 2014)

Audio Synth nabízí čtyři zvukové profily, tři napodobující akustické nástroje (piáno, vathany, kytaru) a jeden EDM elektronickou syntetizaci. Funguje ve všech desktopových a některých mobilních prohlížečích podporujících standard HTML5.[10]

3.4 Audio Player

Celým názvem Audio Player: Responsive & Touch-friendly je plugin pro knihovnu jQuery napsaný v CSS. Toto rozšíření zveřejnil v roce 2012 web designer Osvald Valutis s myšlenkou zůstat co nejblíže nativnímu přehrávači rozšířením tagu *audio* o několik málo základních funkcí. Výsledkem je decentní responsivní přehrávač pro jednotlivé audio soubory stojící jako protipól rozsáhlým pluginům s desítkami funkcí. Autor plugin úspěšně testoval v prohlížečích, které podporují HTML5 audio, Androidu od verze 4.2 a iOSu 6. (Vautis, 2012)

3.5 WexFlow API

VexFlow je webové notační vykreslovací API, napsané kompletně v javascriptu, používající HTML5 canvas a SVG.

WexFlow vyvíjí pod nickem *0xFE* Mohit Muthanna Cheppudira. Prvně jej publikoval v roce 2010, od té doby prochází častými průběžnými aktualizacemi. Dle slov autora (Cheppudira, 2016) by mělo být podporováno všemi moderními

[9] Vychází při tom z materiálů zveřejněných Stanfordskou univerzitou, dostupné z: http://www.acoustics.salford.ac.uk/acoustics_info/sound_synthesis/

[10] Na Androidu 4.4.2 a 6.1 úspěšně otestován v prohlížeči FleshFox, neúspěšně v Chrome a Opera Mini

prohlížeči. Otevřená licence MIT umožňuje téměř libovolné nakládání se zpřístupněnými kódy pod podmínkou uvedení původního copyrightu. (Open Source Initiative, 2006)

Oficiální web se nachází na adrese *VexFlow.com*, zde jsou dostupné základní informace, částečný anglický návod[11] a autorův blog s aktualizacemi a rozšiřujícími informacemi[12]. K diskuzi slouží rozsáhlá podpůrná Google skupina,[13] aktuální zdrojové kódy zpřístupňuje GitHub repositář[14].

V českém prostředí se tohoto tématu dotkla diplomová práce Petra Látala *Webový portál pro výuku hudební teorie* (2011) a články na ni navazující[15], vzhledem k roku vydání zde však narážíme na problém aktuálnosti.

Cheppudira pod štítkem VexFlow vyvíjí i další alternativy pro práci s notací vystavěné právě na VexFlow API, často s velmi podobnými názvy.

V první řadě se jedná o *VexTab*, což je svébytný jazyk pro tvorbu notace a tabulatur určený běžným uživatelům. Ten lze použít buď přímo ve zdrojovém kódu stránky připojením javascriptové knihovny nebo skrze další dílčí aplikace. Pro rychlé zobrazení notového zápisu jazykem *VexTab* slouží jednoduchá aplikace *VexTab Playground*[16], která umožňuje výsledek stáhnout jako obrázek PNG nebo sdílet přes *Imgur*[17]. Dostupná jsou také rozšíření pro dokumenty *Google Docs*[18] a prohlížeč *Google Chrome*[19] a další nástroje.

Pod jménem *My VexFlow*[20] je provozována webová aplikace, která je určena lidem bez programátorské ambice, kteří chtějí snadným způsobem vytvořit hudební ukázku a vložit ji na svůj web vnořeným rámem *iframe*. (Cheppudira, VexFlow) Aplikace je dostupná po registraci zdarma, uživatel v ní notaci zapisuje pomocí jazyka *VexTab*. Bohužel současná verze nepodporuje české znaky, takže v našem prostředí jsou možnosti jejího použití omezené.

---

[11] Chybí například informace o připojení externích souborů, podpoře nativního SVG, návod pro tvorbu více taktů v jednom hudebním úryvku aj. Datum poslední aktualizace není uvedeno, ale dá se předpokládat druhá polovina roku 2014. Dostupné z: http://www.vexflow.com/docs/tutorial.html

[12] Obsahuje zejména novinky a rozšíření, řešení složitějších příkladů. Jedná se o Jedinou pravidelně aktualizovanou část webu. Dostupné z: http://0xfe.blogspot.cz/

[13] Dostupné z: https://groups.google.com/forum/#!forum/vexflow

[14] Dostupné z https://github.com/0xfe/vexflow/tree/master/releases

[15] Např. LÁTAL, P. et al. *Vexflow API in music learning management system*. 2012.

[16] Dostupné z http://www.vexflow.com/vextab/playground

[17] Webové úložiště obrázků určené pro jejich sdílení na webu.

[18] Dostupné z: https://goo.gl/w8V4FV

[19] Dostupné z: https://goo.gl/Bqmwsn

[20] Dostupné z: http://my.vexflow.com/

4 Práce s WexFlow

4.1 Způsob vykreslování

VexFlow API podporuje dva druhy vykreslování notace, vektorový formát SVG a bitmapový HTML5 canvas. Obě technologie zpracovávají tentýž vstupní skript, obě pracují na straně klienta, a na první pohled poskytují i stejný výsledek. Rozdíly najdeme v principu jejich fungování a vhodnosti použití v konkrétní situaci.

Canvas je párový element HTML5, který poskytujíce bitmapové plátno závislé na rozlišení. Nakreslené tvary tedy existují pouze jako pixely, není možné k nim dále přistupovat nebo reagovat na podněty od uživatele.[21] (WhatWG, 2016) Výhodou může být možnost uložit výsledný notový zápis jako obrázek .png nebo .jpg a stálý výkon i při vykreslování rozsáhlejších partů.

SVG je vektorový grafický formát založený na XML. Přizpůsobí se libovolnému rozlišení obrazovky, je proto vhodný pro uživatelská rozhraní, kde je přístupnost klíčová. Ke každému elementu lze přistupovat přes SVG DOM API, což do nich umožňuje zavést interaktivitu. Na displejích s menším rozlišením se při použití SVG na vykreslení notace může projevit mírná neostrost, jedná se ale o velmi subjektivní problém.

Aktuální verze VexFlow (1.2.45) podporuje nativní SVG. Dříve se za tímto účelem připojovala externí knihovna Raphael, toto řešení je sice stále funkční, ale vnímáno jako zastaralé, nové verze pro něj nemají podporu.

Samotný autor (Cheppudira, 2015) doporučuje pro složitější aplikace kombinaci canvasu a nativního SVG.

[21] Respektive by to vyžadovalo manuální překreslení notace.

4.2 Začlenění VexFlow do stránky

VexFlow pro správné fungování potřebuje svou vlastní javascriptovou knihovnu a knihovnu *jQuery*[22]. Oba skripty pro správné fungování připojujeme v hlavičce stránky.

```
...
<head>
       <script src="js/jquery-2.2.3.min.js"> </script>
       <script src=" js/vexflow-min.js"></script>
...
```

Plátno, do kterého se zobrazí notový zápis, umisťujeme do těla stránky, podle zvoleného způsobu vykreslení jako element *canvas* nebo *svg*. Tomuto elementu je nutné nastavit velikost odpovídající obsahu[23] a identifikátor.

```
<canvas id="mojeid" width="700" height="100"></canvas>
```

```
<svg id="mojeid" width="700" height="100" viewbox = "0 0 width height"> </svg>
```

4.3 Vykreslení notové osnovy

Pro vykreslení prázdné notové osnovy začneme zápis jedním z následujících skriptů v závislostí na tom, jestli pracujeme s canvasem nebo svg.

```
var renderer = new Vex.Flow.Renderer(mojeid, Vex.Flow.Renderer.Backends.CANVAS);
```

```
var renderer = new Vex.Flow.Renderer(mojeid, Vex.Flow.Renderer.Backends.SVG);
```

Na prvním řádku voláme objekt s identifikátorem *mojeid* a tzv. backend (rozeznání typu dat). Poté použijeme funkci *Vex.Flow.Renderer ()* pro vygenerování kontextu pro notovou osnovu s parametry odkazovaného plátna a uložíme do proměnné *render*.

[22] Aktuální verze dostupná z: https://jquery.com/download/
[23] V případě plátna menšího, než je zobrazovaný obsah, canvas notovou osnovu nezobrazí vůbec, svg jen její část.

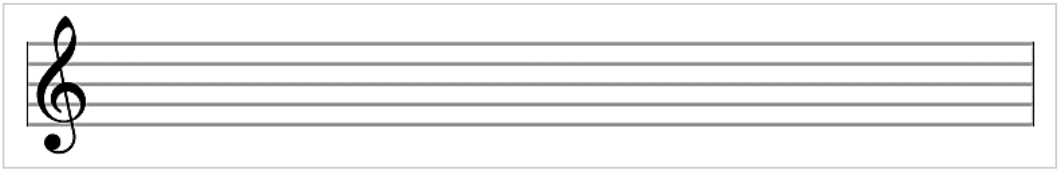
V dalším kroku vygenerujeme notovou osnovou a houslovým klíčem:

```
var ctx = renderer.getContext();
var stave = new Vex.Flow.Stave(10, 0, 500);
stave.addClef("treble").setContext(ctx).draw();
```

Na prvním řádku vidíme proměnnou *ctx*, ta dokončuje rozhraní pro 2D kreslení odpovídající HTML5 canvas nebo SVG a definuje ji výsledek funkce *getContext()*.

V proměnné *stave* vytvoříme novou notovou osnovu pomocí funkce *Vex.Flow.Stave().* Parametry této funkce zadáme (postupně) v pixelech délku notové osnovy, její pozici zleva a shora. Pro pozici se zpravidla zadávají nuly, nedefinují ale přesné umístění osnovy, protože VexFlow si nechává určitou prostorovou rezervu pro vyšší noty. (Cheppudira, The VexFlow Tutorial) Případné odsazení z designových důvodů lze efektivněji řešit kaskádovými styly. Délku notové osnovy je vhodné nastavit o několik pixelů menší než vlastnost *width* prvku, do kterého bude vykreslena, aby nedocházelo k zakrytí

K osnově přidáme houslový klíč příkazem *addClef("treble").* Dostupných je 22 variant klíčů.[24] Další často používaný, basový klíč, můžeme vložit příkazem *addClef("bass").* Vše vykreslíme pomocí *draw()*.



Obrázek 5 – VexFlow, vykreslení prázdné notové osnovy s houslovým klíčem

Celý zdrojový kód pro základní výpis prázdné notové osnovy s basovým klíčem včetně výsledného zobrazení obsahuje příloha č. 1.

4.4 Model vykreslování not

Pro vkládání not je nejprve třeba pochopit model vykreslování dat. Jde o složitější problematiku i proto, že systém se nemusí bez znalosti hudební teorie zdát na první pohled zcela logický.

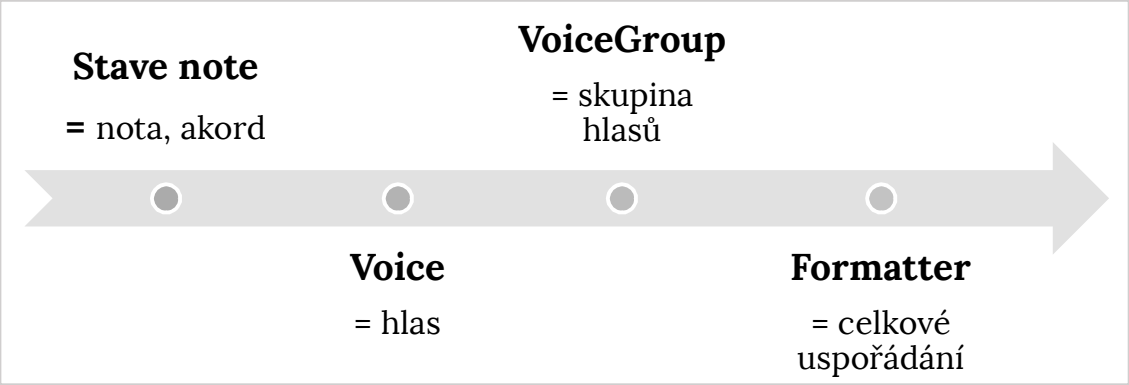
[24] Kompletní seznam klíčů dostupný z: http://www.vexflow.com/docs/clef.html

Základním prvkem je *StaveNote*, nota nebo skupina not pomyslně náležící k jedné notové nožce. Může jít samotnou notu i akord, podstatné však je, že všechny tato noty patří do stejného hlasu, začínají na tutéž dobu a mají stejnou délku.

Tyto *StaveNote* se sdružují do *Voice* (v češtině hlasů). Ty jsou nositeli časové signatury (taktu) a provádí kontrolu, zda počet *StaveNote* v nich obsažených odpovídá předepsané hodnotě. Pokud někde vyhodnotí chybu (např. pouze tři čtvrťové noty v 4/4 taktu), kód není validní a vypíše se pouze prázdná osnova.

Hlasy *Voice* se dále sdružují do *VoiceGroup*. Tento prvek získává na významu v situacích, kdy notace obsahuje více hlasů – zajistí každé notě správnou orientaci nožičky nahoru nebo dolů podle hlasu, do kterého náleží. V opačném případě jde o pouhý obal jednoho hlasu bez zvláštní funkce. Aby byl kód validní, musí *VoiceGroup* obsahovat alespoň jeden *Voice*.

Nejvýše v hierarchii stojí *Formatter*, který zajišťuje správné rozložení not na notové osnově podle předepsaných pravidel. (Cheppudira, The VexFlow Tutorial)



Obrázek 6 - Model vykreslování prvků ve VexFlow API
(vlastní zpracování dle Cheppudira, The VexFlow Tutorial)

4.5 Základní vkládání not a pomlk

Konkrétní notu vytváříme pomocí funkce *Vex.Flow.StaveNote*, každý tón je v zápisu určen dvěma vlastnostmi, výškou a délkou. V případě akordu zapisujeme několik výšek tónů, ale pouze jednu délku pro celou skupinu.

```
new Vex.Flow.StaveNote({ keys: ["d/5"], duration: "4" })      // tón d2 čtvrťový
```

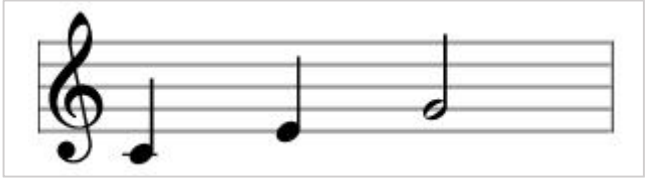
```
new Vex.Flow.StaveNote({ keys: ["c/4", "e/4", "g/4"], duration: "8" })
       // kvintakord C Dur – tóny c1, e1 a g1 osminové
```

Délku noty definuje vlastnost *duration*. Zapisujeme ji celým číslem, konkrétně jmenovatelem původní hodnoty. Například číslo 4 znamená notu čtvrťovou, číslo 8 notu osminovou a číslo 1 notu celou.

Výšku tónu definuje vlastnost *keys*. Tu zapisujeme jako jméno tónu a označení oktávy oddělené lomítkem. Používá se anglosaské hudební názvosloví, takže tón *h* se zapisuje jako *b* a jiné názvy se používají také pro oktávy. Náš tón (malé) *c* zde odpovídá označení *c/3*, tón *c1* zapisujeme jako *c/4*, tón *c2* jako *c/6* apod.

Následující ukázka vypisuje první takt písničky Ovčáci, čtveráci. Jde o jednoduchý takt o dvou čtvrťových a jedné půlové notě.

```
var renderer = new Vex.Flow.Renderer(mojeid,Vex.Flow.Renderer.Backends.CANVAS);

var ctx = renderer.getContext();
var stave = new Vex.Flow.Stave(0, 0, 260);
stave.addClef("treble").setContext(ctx).draw();

// Vytvoří noty
var notes = [
new Vex.Flow.StaveNote({ keys: ["c/4"], duration: "4" }),  // nota c1 čtvrťová
new Vex.Flow.StaveNote({ keys: ["e/4"], duration: "4" }),  // nota e1 čtvrťová
new Vex.Flow.StaveNote({ keys: ["g/4"], duration: "2" }),  // nota g1 půlová
];
// Vytvoří hlas ve 4/4 taktu
var voice = new Vex.Flow.Voice({
num_beats: 4,
beat_value: 4,
resolution: Vex.Flow.RESOLUTION
});
// Přiřadí noty do hlasu
voice.addTickables(notes);

// Upraví rozložení not pro 260px
var formatter = new Vex.Flow.Formatter().
joinVoices([voice]).format([voice], 250);

// Vykreslí hlas
voice.draw(ctx, stave);
```



Obrázek 7 – VexFlow, vykreslení jednoduchého taktu bez předznamenání

Pomlku tvoříme jako běžnou notu, ale za její délku uvedeme písmeno *r* (z anglického *rest*). Automatické doplnění posazení na osnově zajistí syntaxe *b/4*, manuálně lze výšku pozice znaku nastavit stejným písmenným a číslicovým systémem jako výšku noty.

```
new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "8r" }),      // pomlka osminová
new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "2r" }),      // pomlka půlová
new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "1r" }),      // pomlka celá
```

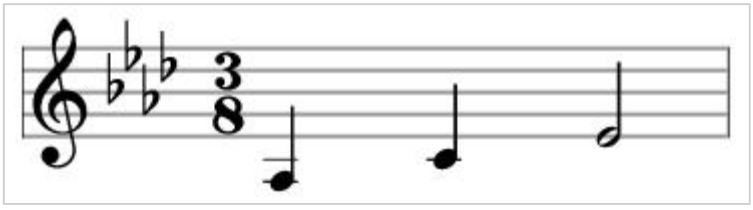
Posuvky a tečky u not se zapisují jako zvláštní znaky, věnuje se jim kapitola 4.7.

4.6 Značení tóniny a taktu

Pro udání tóniny a taktu na začátku notace (za notovým klíčem) slouží dvě snadno použitelné funkce *addKeySignature()* a *addTimeSignature()*. Vykreslení osnovy *draw()* je oproti předchozím případům třeba vyčlenit samostatně a posunout až za tyto funkce.

Podívejme se na rozšířený příklad vykreslení prázdné notové osnovy, tentokrát s předznamenáním As dur a značením 4/4 taktu:

```
var renderer = new Vex.Flow.Renderer(mojeid,Vex.Flow.Renderer.Backends.CANVAS);

var ctx = renderer.getContext();
var stave = new Vex.Flow.Stave(0, 0, 320);
stave.addClef("treble").setContext(ctx);          // vloží houslový klíč
stave.addKeySignature("Ab").setContext(ctx);      // vloží předznamenání As dur
stave.addTimeSignature("4/4").setContext(ctx);    // vloží značení taktu 4/4
stave.draw();                                     // vykreslí osnovu
```



Obrázek 8 – VexFlow, vykreslení taktu s předznamenáním a taktovým označením

Značení taktu ukládáme do *TimeSignature* v obecně užívané podobě (např. 3/8 pro tříosminový takt) a předznamenání *KeySignature* jako název tóniny v anglosaském hudebním názvosloví (a to pouze tónem bez přívlastku minor/major). Při vložení časového označení a předznamenání s větším počtem posuvek je vhodné nastavit šířku osnovy *stave* o několik desítek pixelů větší než prostor pro výpis not *formatter*, tím lze předejít vykreslení poslední noty mimo osnovu.

Tabulka 1 - České a anglosaské značení durových tónin

	7♭	**6♭**	**5♭**	**4♭**	**3♭**	**2♭**	**1♭**	**0**	**1#**	**2#**	**3#**	**4#**	**5#**	**6#**	**7#**
ČJ	Ces	Ges	Des	As	Es	B	F	C	G	D	A	E	H	Fis	Cis
AJ	Cb	Gb	Db	Ab	Eb	Bb	F	C	G	D	A	E	B	F#	C#

Je důležité si uvědomit, že funkce *addKeySignature()*, *addTimeSignature()* a *addClef()* pouze vypisují daný grafický znak, ale nijak neovlivňují další obsah. Vlastní notový zápis se tvoří vždy stejným způsobem bez ohledu na klíč, tóninu nebo značení taktu. Pro počítání dob v taktu jsou směrodatné hodnoty v proměnné *voice*.

4.7 Modifikátory: tečky a posuvky

Posuvky, tečky a některá další znaménka jsou ve VexFlow API realizovány pomocí tzv. modifikátorů, které v podstatě slouží k obohacení not různými dalšími znaky. Všechny modifikátory patřící k jedné notě nebo akordu společně tvoří *ModifierContext*, který jim umožňuje vzájemně na sebe reagovat, například upravit umístění tak, aby se ve výpisu notace nepřekrývaly. (Cheppudira, The VexFlow Tutorial)

Zápis noty s tečkou provedeme tak, že za údaj o její délce připíšeme písmeno *d* (z anglického *dot*), tím jednak změníme její notovou délku pro sčítání dob v taktu, a zároveň určíme přesnou pozici tečky. Následně voláme funkci *addDotToAll()* pro vykreslení této tečky. Zde je vhodné upozornit na změnu syntaxe, první řádek končí tečkou, nikoli čárkou jako v předchozích příkladech.

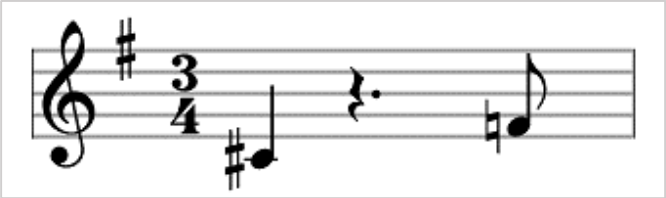
```
// nota g1 čtvrťová s tečkou
new Vex.Flow.StaveNote({ keys: ["g/4"], duration: "4d" }).
       addDotToAll(),                              // přidání grafického znaku tečky
```

Stejný postup můžeme aplikovat na pomlku s tečkou. Drobný rozdíl spočívá v tom, že po udání délky následuje písmeno *d* (tečka) a až za ním písmeno *r* (pomlka).

```
// pomlka půlová s tečkou
new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "2dr" }).
       addDotToAll(),                              // přidání grafického znaku tečky
```

Modifikátory používají také pro zobrazení posuvek. Příslušný počet křížků nebo béček uvedeme v názvu noty (např. *fis* jako *f#*, *eses* jako *ebb*), v případě odrážky ponecháme původní hodnotu. Poté vykreslíme daný symbol vyvoláním funkce *addAccidental()*, v jeho uvozovkách opět uvedeme požadovanou značku - křížek *#*, dvojkřížek *##*, béčko *b*, dvojbéčko *bb* nebo odrážku *n*.

```
// nota cisis1 půlová
new Vex.Flow.StaveNote({ keys: ["c##/4"], duration: "2" }).
       addAccidental(0, new Vex.Flow.Accidental("##")),
       // přidání grafického znaku křížku
```



Obrázek 9 - VexFlow, vykreslení taktu s modifikátory

Pokud jsou posuvky použity v akordech, noty a znaménka, které k sobě patří, identifikujeme podle jejich zástupné číselné hodnoty. Ta se automaticky přiřazuje od první zapsané noty, která má hodnotu nula. Ve funkci *addAccidental()* poté touto hodnotou adresujeme konkrétní notu, a tak získáme jistotu správného přiřazení.

```
// dominantní septakord A7 čtvrťový
new Vex.Flow.StaveNote({ keys: ["ab/4", "c/5", "eb/5", "gb/5"], duration: "4" }).
   addAccidental(0, new Vex.Flow.Accidental("b")).   //b pro první notu v pořadí
   addAccidental(2, new Vex.Flow.Accidental("b")).   //b pro třetí notu v pořadí
   addAccidental(3, new Vex.Flow.Accidental("b")),   //b pro čtvrtou notu v pořadí
```

Zdrojový kód složitější notační ukázky s modifikátory, předznamenáním a časovou signaturou je obsažen v příloze č. 2.

4.8 Zápis více taktů

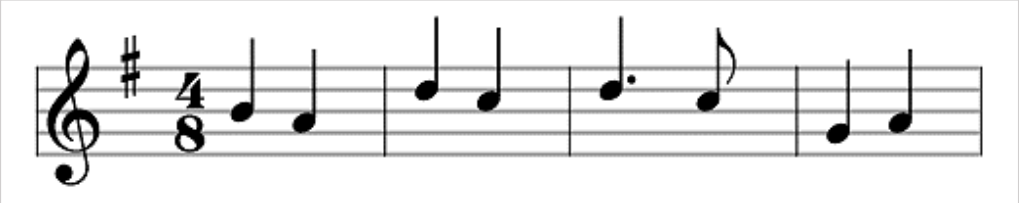
Zápis více taktů je jednou z ne zcela dořešených částí VexFlow. Původní návrh nepočítal s tím, že by jedna osnova obsahovala více jak jeden takt. (Cheppudira, 2011) V současnosti jsou dostupné dva způsoby tvorby taktů, ani jeden se nedá považovat za zcela ideální řešení.

Oficiální cesta vychází z původního návrhu. Pro každý takt se vytváří vlastní osnova *Stave*, každá další osnova je horizontálně posunuta a umístěna vedle předchozí pomocí juxtapozice. (Cheppudira, Greg, 2011) Tento přístup je sice komplikovanější, ale umožňuje použití některých speciálních funkcí jako *setBegBarType()* a *setEndBarType()* pro zvolení typu taktové čáry.

Tabulka 2 – Druhy taktových čar podporovaných VexFlow API
(vlastní zpracování podle VexFlow dokumentace)

SINGLE	**DOUBLE**	**END**	**REPEAT_BEGIN**	**REPEAT_END**	**NONE**
jednoduchá	dvojitá	ukončovací	začátek repetice	konec repetice	žádná

Převzatý příklad zápisu více taktů pomocí juxtapozice s doplněným českým komentářem obsahuje příloha č. 3.



Obrázek 10 – VexFlow, vykreslení více taktů

Druhá metoda vznikla původně jako jakýsi „hack“, ale pro svou přívětivost zůstává mezi uživateli VexFlow stále frekventována.[25] Spoléhá na vypnutí počítání not v taktu příkazem *voice.setStrict(false)* a vložení samostatných taktových čar na vybraná místa funkcí *BarNote()*. Vzniká tak v podstatě jeden velký takt s přidanými grafickými symboly, ale výsledné vykreslení poskytuje stejný výsledek jako první způsob. (Cheppudira, Greg, 2011)

[25] Na jejím principu staví i aplikace vytvářená v této bakalářské práci.

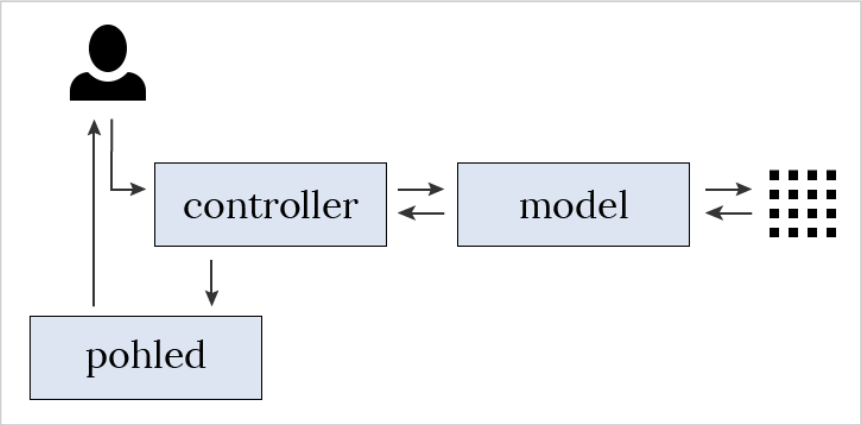
V následující ukázce jsou tímto způsobem vypsány první tři takty písničky *Kočka leze dírou*.

```
var renderer = new Vex.Flow.Renderer(canvas, Vex.Flow.Renderer.Backends.CANVAS);

    var ctx = renderer.getContext();
    var stave = new Vex.Flow.Stave(10, 0, 750);
    stave.addClef("treble").setContext(ctx).draw();

    var notes = [
    new Vex.Flow.StaveNote({ keys: ["c/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["d/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["e/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["f/4"], duration: "8" }),

    new Vex.Flow.BarNote(),         // vložení taktové čáry

    new Vex.Flow.StaveNote({ keys: ["g/4"], duration: "4" }),
    new Vex.Flow.StaveNote({ keys: ["g/4"], duration: "4" }),

    new Vex.Flow.BarNote(),         // vložení taktové čáry

    new Vex.Flow.StaveNote({ keys: ["a/4"], duration: "4" }),
    new Vex.Flow.StaveNote({ keys: ["a/4"], duration: "4" }),
    ];

    var voice = new Vex.Flow.Voice({num_beats: 4, beat_value: 4,
        resolution: Vex.Flow.RESOLUTION
    });
    voice.setStrict(false)          // Vypnutí počítání dob v taktu

    voice.addTickables(notes);

    var formatter = new Vex.Flow.Formatter().
    joinVoices([voice]).format([voice], 700);

    voice.draw(ctx, stave);
```

5 Návrh a implementace aplikace

5.1 MVC architektura

Kód aplikace staví na principu objektově orientovaného programování (OOP) a architektuře MCV (*Model-View-Controller*). Dělí se do tří samostatných logických vrstev, přičemž změny v jedné nevyžadují výrazný zásah i do částí dalších. To je značnou výhodou vhledem k modularitě aplikace a předpokladům budoucího rozšíření projektu.



Obrázek 11 - MVC model (vlastní zpracování)

Kontroler *(controller)* má na starosti tok událostí, přijímá požadavek uživatele (před HTTP GET a POST nebo interakci s UI), podle kterých volá a zpracovává data s příslušného modelu a posílá je do pohledu *(view)*. Řídí celkovou aplikační logiku, v podstatě plní organizační funkci. O komunikaci s databází se stará model, vyžádaná data předává zpět kontroleru. Tyto dvě vrstvy pracují nezávisle na šabloně, která se zobrazí uživateli. (Pastor, 2010)

Kontrolery a modely aplikace jsou společně uloženy do složky *class/*, přičemž každá skupina tříd má svou podsložku. Jelikož jsou úzce propojeny, tento systém hledící i na obsah mi přišel přehlednější než jejich striktní oddělení do různých složek.

Pohled je v podstatě šablonou pro zobrazení dat, přijímá potřebná data z kontroleru a vytváří UI na straně klienta. Všechny pohledy aplikace jsou ukládány ve složce *template/cs/*. Aplikace je připravená na vícejazyčné rozhraní - stačí vytvořit další složku s pohledy, např. *en/*, a předávat kontroleru zvolený jazyk jako hodnotu proměnné.

5.2 Návrh databáze

Samotnému programování předcházel návrh poměrně složité databáze. V té bylo třeba ukládat jednotlivé taktové úryvky v takové podobě, aby aplikace disponovala dostatkem informací k vyhodnocení jejich intonační a rytmické obtížnosti, správnému napojování a transpozicím do jiné tóniny. Také musela obsahovat vodítka k přesnému znění a notačnímu výpisu jednotlivých taktů, respektive řešit situaci, kdy více not (druhů grafického zápisu) má stejnou tónovou výšku (zní se stejné frekvenci) a jejich výběr podléhá dalším zákonitostem.

RM diagram navrhované databáze obsahuje příloha č. 4.

Obecné vlastnosti každého vloženého taktu obsahuje tabulka *takty*. K jejímu primárnímu klíči *id_taktu* se váže několik referencí. Dále obsahuje informaci o taktovém značení (ta je rozdělena do sloupců *délka_doby* a *počet_dob*, aby mohla být zapsána číselným typem). Pokud takt splňuje podmínky pro umístění na konci v intonačního cvičení, sloupec *poslední_takt* nabývá hodnoty 1.

Tabulka 3 – Návrh databáze, tabulka *takty*

id_taktu int(11), AI, PK	**delka_doby** tinyint(4)	**pocet_dob** tinyint(4)	**posledni_takt** tinyint(4)
1	2	4	0
2	2	4	0
3	3	8	1

Pro získání maximální variability cvičení jsem se rozhodla pro ukládání taktů vždy v základní tónině (C dur), následnou transpozici umožní číselné klasifikace v tabulce *tóny* a údaje o poloze a předznamenání v tabulce *toniny* a *tony_tonin*.

Obsah taktu zaznamenává tabulka *noty*. Zde každý řádek obsahuje vlastnosti jedné konkrétní noty umístěné v některém z taktů. Notě je přiřazeno ID a reference na *id_taktu*, do kterého náleží. Sloupec *pozice* vypovídá o pořadí v taktu a *delka* je jmenovatelem rytmické hodnoty (např. 4 pro čtvrťovou). Pokud se jedná o notu *s_teckou*, nabývá hodnoty 1. Reference *id_tonu* odkazuje na tabulku *tony*, její popis následuje.

Tabulka 4 - Návrh databáze, tabulka *noty*

id_noty int(11), AI, PK	**id_taktu** int(11)	**pozice** tinyint(4)	**id_tonu** int(11)	**delka** smallint(6)	**s_teckou** tinyint(4)
1	1	1	6	4	0
2	1	2	7	2	1
3	2	1	43	8	0

Informace o výšce a grafické podobě jednotlivých tónů obsahuje tabulka *tony*. Kromě primárního klíče *id_tonu* zde najedeme sloupec *jmeno* a *oktava* (zde pro zjednodušení používám anglosaské značení platné ve VexFlow API a použitém zvukovém syntetizátoru). *Posuvky* udávají jaká znaménka a v jakém počtu by měla být k danému tónu vykreslena. A nakonec *hodnota*, která definuje přesnou zvukovou výšku tónu a bude využita pro transpozice do jiných tónin.

Tabulka 5 - Návrh databáze, tabulka *tony*

id_tonu int(11), AI, PK	**jmeno** varchar(4)	**oktava** tinyint(4)	**posuvky** tinyint(4)	**hodnota** tinyint(4)
1	ab	3	-1	33
2	c	4	0	37
3	c#	4	1	38

Určení intonační obtížnosti závisí na počtu a kvalitě volných nástupů. Ty mohou být jednak v průběhu taktu, jednak na jeho začátku, pokud se poslední nota předchozího taktu a první následujícího liší o interval tercie a větší. Proto jsou v tabulce *volne_nastupy* ukládány informace o průběžných volných nástupech a vždy prvním tónu každého taktu (který se volným nástupem stát může a nemusí). Tabulka obsahuje primární klíč, referenci *id_taktu*, *stupen* v podobě jeho číselného označení (spodní stupně jsou ukládány se znaménkem -) a informaci, zda jde o *prvni_ton*. Na základě dat této tabulky jsou takty vybírány pro konkrétní cvičení podle intonační obtížnosti zvolené uživatelem.

Tabulka 6 - Návrh databáze, tabulka *volne_nastupy*

id_nastupu int(11), AI, PK	**id_taktu** int(11)	**stupen** tinyint(4)	**prvni_ton** tinyint(4)
1	1	1	1
2	2	4	1
3	2	-6	0

Otázku transpozic řeší tabulky *toniny* a *tony_tonin*. První jmenovaná obsahuje primární klíč *id_toniny*, její *nazev* podle anglosaského názvosloví (používá se pro vykreslení předznamenání ve VexFlow) a číselnou informaci o jejím *predznamenani* (od -7 pro béčka do 7 pro křížky) Ve sloupci *moll* se ukládá údaj o tom, zda se jedná o tóninu v dur nebo moll (nabývá hodnoty 0 nebo 1). Směrodatnou hodnotou pro transpozici je *posun*, který udává, o kolik výškových hodnot a jakým směrem se mají noty vybraných taktů posunout. Dále udává maximální a minimální tónovou výšku v základní tónině, kterou může aplikace vybrat, aby byla pro uživatele po transpozici zpívatelná (sloupce *vyska_max* a *vyska_min*).

Tabulka 7 - Návrh databáze, tabulka *toniny*

id_toniny int(11), AI, PK	**nazev** varchar(2)	**moll** tinyint(1)	**predzn.** tinyint(4)	**posun** tinyint(4)	**vyska_max** tinyint(4)	**vyska_min** tinyint(4)
1	G	0	1	7	27	44
2	Ab	0	-4	-4	37	56
3	C	0	0	0	34	51

Tabulka *tony_tonin* obsahuje seznam všech tónů dané tóniny, kromě primárního klíče ji určuje *id_toniny*, *id_tonu* a *hodnota_tonu*. V podstatě doplňuje souvislost, pro jaký údaj *hodnota_tonu* (odpovídá sloupci *hodnota* v tabulce *tony*) má být v dané tónině zvoleno které grafické vykreslení odpovídající *id_tonu* v tabulce *tony*.

Tabulka 8 - Návrh databáze, tabulka *tony_tonin*

id_zaznamu int(11), AI, PK	**id_toniny** int(11)	**id_tonu** int(11)	**hodnota_tonu** tinyint(4)
1	1	4	34
2	1	5	36
3	1	6	37

Další tabulky stojí stranou hlavní funkcionality aplikace.

Při řešení problému duplicit taktů se jako nejvhodnější řešení jevilo vytvoření unikátního klíče každého taktu na základě not v něm obsažených a jeho taktového značení (sloupec *klic*). Vzhledem k tomu, že tabulka *takty* již byla využívána na mnoha místech aplikace, a její úprava by proto vyžadovala velký zásah do kódu,

jako nejvhodnější řešení se zdálo vytvoření další tabulky *klice_taktu* obsahující referenci *id_taktu* na tabulku *takty*.

Tabulka 9 – Návrh databáze, tabulka *klice_taktu*

id_klice int(11), AI, PK	**id_taktu** int(11)	**klic** varchar(255)
1	1	2441480128013801180
2	2	242740640
3	4	3451480151601616015401440

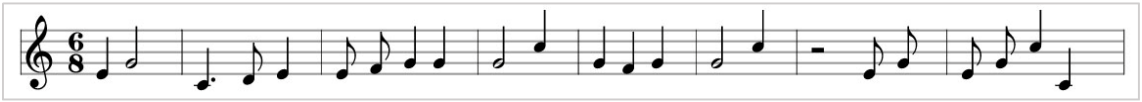
Pomocná tabulka *operne_pisne* uchovává záznamy pro šablonu *pisen*. Vedle primárního klíče a číselného údaje *stupen* jsou to sloupce *tonina*, *prvni_ton* a *pisen*, které používám jako odkaz na zvukové a obrázkové soubory téhož jména.

Tabulka 10 – Návrh databáze, tabulka *operne_pisne*

id_pisne int(11), AI, PK	**stupen** tinyint(4)	**tonina** varchar(2)	**první_ton** varchar(2)	**pisen** varchar(16)
1	-7	F	e1	Koukej
2	-5	G	d1	Zima
3	1	C	c1	Kocka

5.3 Výpis intonačního cvičení

Na výpisu intonačního cvičení podle uživatelských kritérií se podílí několik objektů, jejich metod a funkcí, vybrané z nich popisují následující podkapitoly. Ukázky jejich zdrojového kódu jsou obsaženy v příloze č. 7 a 8.

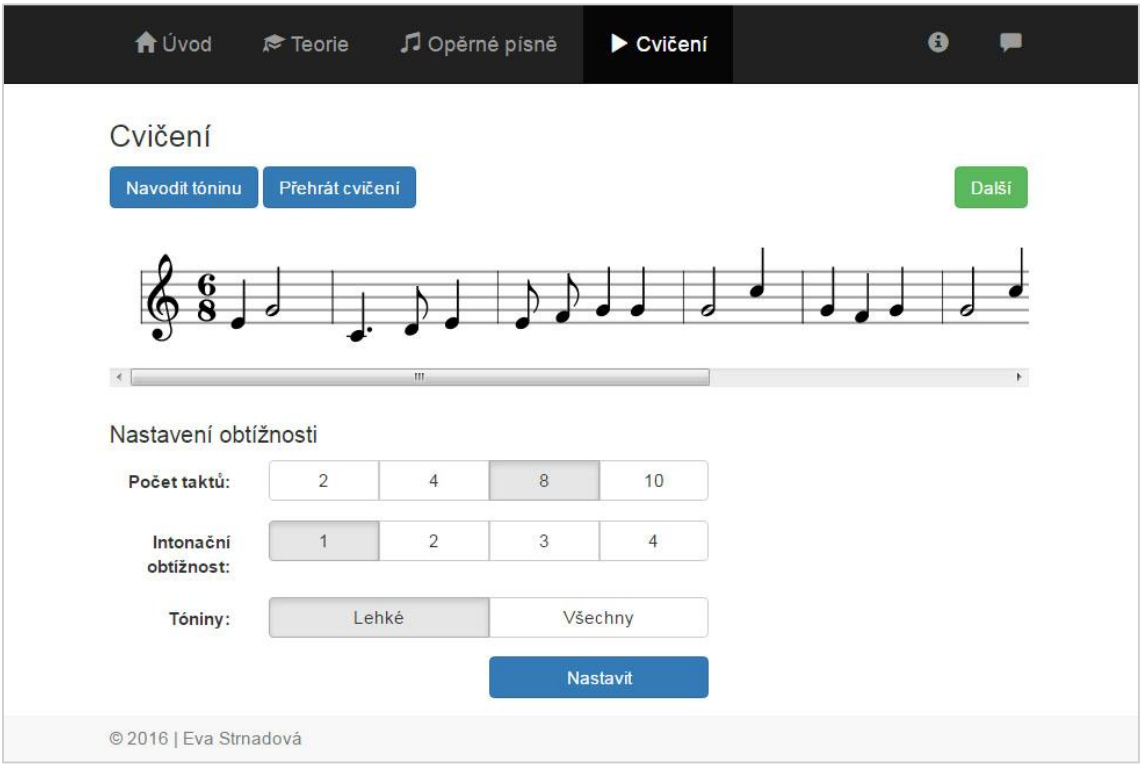


Obrázek 12 – Uživatelské rozhraní aplikace, příklad výpisu intonačního cvičení

5.3.1 Princip výpisu cvičení

Uživatel dává vstupem do záložky „Cvičení“ přes URL pokyn k inicializaci kontroleru *cviceni*. Na stránce se mu zobrazí notový úryvek vygenerovaný podle předdefinovaných hodnot s možností navození tóniny a přehrání. Může pokračovat tlačítkem další nebo pod cvičením nastavit požadovanou úroveň podle třech

parametrů: počet taktů, intonační a tonální obtížnost. Systém je dále připraven pro rozšíření o volbu rytmické obtížnosti.



Obrázek 13 – Uživatelské rozhraní aplikace, cvičení (zobrazení pro menší displeje)

Kontroler *cviceni,* v první řadě zjišťuje, zda byl uživatelem odeslán formulář *nastaveni_obtiznosti*, pokud ano, převede data z odeslaného formuláře na své vlastnosti, pokud ne, použije předdefinované hodnoty. Následně si do proměnné *model* načte objekt *cviceni*, kterému předá požadovaný počet taktů, intonační obtížnost atd. a jako pohled zvolí šablonu *cviceni*.

Model *cviceni* si při své inicializaci uloží všechny předané proměnné a na základě jejich hodnot vytvoří objekty *rytmicka_obtiznost*, *intonacni_obtiznost* a *toninova_obtiznost*, které se v hlubší části kódu starají o výběr taktů odpovídajících požadavkům. Na závěr vytvoří objekt *notace*, který je v podstatě jádrem samotného výpisu, ten za pomoci předaných objektů vybere a seřadí vhodné takty a vytvoří z nich ucelené pole.

V šabloně *cviceni* poté na konkrétním místě volá kontroler funkce dvou pomocných statických tříd *notacePlayer* a *notacePresenter*. První z nich má za úkol vytvořit pomocí javascriptu syntetizovanou melodii, druhá třída se stará o samotný výpis notace. Oběma je předán jako parametr objekt *notace* uložený v modelu *cviceni*.

5.3.2 Intonační obtížnost

Uživatel má možnost zvolit si požadovanou intonační obtížnost ve čtyřech úrovních, které jsou sestaveny v souladu Danielovou metodologií. Třída *intonacniObtiznost* se volá při výběru každého taktu zvlášť, protože záleží na jeho pořadí ve cvičení. Řídicí struktura switch definuje pro každou obtížnost hodnoty žádoucích a nežádoucích stupňů, ty jsou následně aplikovány na výběr. *Nastupy_zakazane* slouží k vyřazení některých volných nástupů, které jsou pro začátečníky příliš obtížné a v taktu nesmí být vůbec obsaženy. *Nastupy_cilene* naopak zajišťují obsah pro těžší obtížnosti, aby nemohl nastat případ, kdy aplikace náhodně ze všech nenačte pouze lehké takty. Klíč pole značí, v kolikátém taktu má být některý z daných stupňů zastoupen.

```
public function __construct($obtiznost) {
   switch($obtiznost) {
   case '1':
       $this->nastupy_zakazane = array(-5, -6, -7, 2, 4, 6, 7, 9, 10, 11, 12);
       break;
   case '2':
       $this->nastupy_zakazane = array(-6, -7, 4, 7, 9, 10, 11, 12);
       break;
   case '3':
       $this->nastupy_zakazane = array(-6, 9, 10, 11, 12);
       $this->nastupy_cilene[4] = array(-5, -7, 2, 4, 6);
       break;
   case '4':
       $this->nastupy_zakazane = array();
       $this->nastupy_cilene[2] = array(-5, -7, 7);
       $this->nastupy_cilene[6] = array(-5, -7, 7);
       $this->nastupy_cilene[10] = array(-5, -7, 7);
       $this->nastupy_cilene[4] = array(2, 4, 6);
       $this->nastupy_cilene[8] = array(2, 4, 6);
       break;
   default:
       break;
   }
}
```

5.3.3 Tonální obtížnost a transpozice

Tonální obtížnost nabízí uživateli na výběr ze dvou možností, a to buď ze všech tónin nebo pouze z „jednodušších“ omezených předznamenáním do 3 křížků nebo béček.

Jak bylo uvedeno v kapitole 5.2, v databázi jsou takty ukládány pouze v základní tónině C dur a předpokládá se jejich transpozice do ostatních tónin. Důvodem toho řešení byla snaha o co největší variabilitu cvičení.

Funkcionalitu transpozice volá hned ve svém úvodu objekt *notace*. Ten vyžádá po třídě *toninovaObtiznost* načtení náhodné tóniny z dané kategorie a podle ní i vyřazení výškově nevhodných taktů načtených z databáze. Její funkce *redukujTakty()* na základě zvolené tóniny a jejího omezení maximální a minimální hodnoty tónu vyřadí z pole taktů ty, které neodpovídají požadavkům.

```
public function redukujTakty($takty) {
     $redukovane_takty = array();
     $pocet = count($takty);
     for($i = 0; $i < $pocet; $i++) {
        for($j = 0; $j < count($takty[$i]->noty); $j++) {
           if($takty[$i]->noty[$j]->ton->hodnota < $this->tonina->vyska_min
              AND $takty[$i]->noty[$j]->ton->hodnota != 0) {
                unset($takty[$i]);
                break;
           }
           if($takty[$i]->noty[$j]->ton->hodnota > $this->tonina->vyska_max) {
                unset($takty[$i]);
                break;
                }
        }
     }
     foreach($takty as $takt)  {$redukovane_takty[] = $takt;}
     return $redukovane_takty;
    }
```

Samotná transpozice probíhá až těsně před výpisem, kdy je na každém objektu *takt* volána jeho metoda *transponuj*. Každý takt posune hodnotu všech svých tónů o podle vlastnosti tóniny *posun* a následně porovnává jednotlivé tóny s uloženými tóny dané tóniny - pokud hodnotu najde, vyvolá v objektu *nota* metodu *zmenTon*.

Příslušná nota následně změní svůj objekt *ton* podle ID tónu uloženého mezi tóny dané tóniny.

```
public function transponuj($tonina) {
        for($i = 0; $i < count($this->noty); $i++) {
            $hodnota = $this->noty[$i]->ton->hodnota + $tonina->posun;
            for($j = 0; $j < count($tonina->tony); $j++) {
                if($tonina->tony[$j]->hodnota == $hodnota) {
                    $this->noty[$i]->zmenTon($tonina->tony[$j]->id_tonu);
                }
            }
        }
}
```

```
public function zmenTon($id_tonu) {
        if($this->ton->id_tonu != 0) {
            $this->ton = new Ton\Ton($id_tonu);
        }
    }
```

5.3.4 Úprava grafické podoby výpisu

O výpis notace se stará *NotacePresenter*, který zpracovává vybrané takty do podoby popsané v kapitole č. 4 *Práce s VexFlow API*. Drobným prvkem, na který bych zde chtěla upozornit, je statická funkce *urciRozmery*.

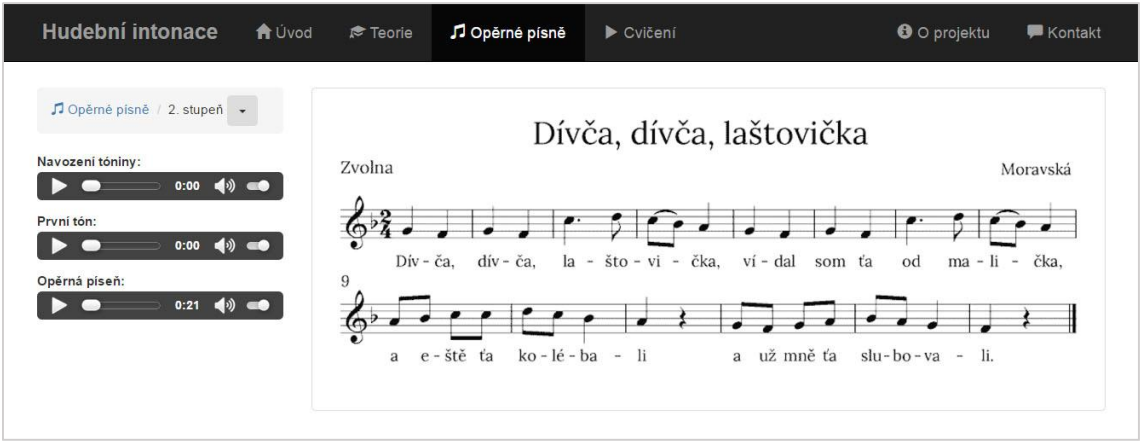
```
static function urciRozmery($notace) {
        $pocet = 0;
        foreach ($notace->takty as $takt) {
            foreach ($takt->noty as $nota) {
                $pocet++;
            }
        }
        $sirka = 35 * $pocet;
        $sirka = $sirka + (count($notace->takty) * 15);
        $sirka = $sirka + 50;
        return array ('vyska' => '110', 'sirka' => $sirka);
    }
```

Při tvorbě aplikace jsem narazila na problém velkých rozdílů délky jednotlivých cvičení. Protože VexFlow API neumí automaticky nastavit šířku svého plátna ani délku vypisované notové osnovy, tyto údaje se musí zapisovat manuálně. Funkce *urciRozmery* přičítá rezervu pro notový klíč a předznamenání k dynamické hodnotě stanovené na základě počtu taktů a not v nich obsažených.

5.4 Výpis opěrných písní

Trochu upozaděna za hlavním obsahem aplikace, ale neméně důležitá zůstává část webu sloužící k seznámení s opěrnými písněmi. Uživatel si požadovaný stupeň vybírá v menu, stupně jsou zde přehledně barevně vyznačeny podle obtížnosti.

Po zvolení stupně (a jemu odpovídající URL) šablona načte podle hodnot uložených v tabulce *operne_pisne* obrazové a zvukové soubory. Ke každé písni náleží zvukový záznam malé kadence k navození tóniny, počátečního tónu a celé lidové písně. Zůstává výhodou online prostředí, že uživatelé mohou při nácviku postupovat vlastním tempem a k písni se kdykoli podle potřeby vrátit.



Obrázek 14 – Uživatelské rozhraní aplikace, opěrná píseň pro 2. stupeň

Závěr

Téma této bakalářské práce, multimediální aplikace pro výuku intonace, bylo v minulosti v České republice několikrát teoreticky otevřeno, ale nikdy nedočkalo praktické realizace. O to aktuálnější se zdál požadavek najít k tématu nový přístup a posunout jej dále v rovině informatické.

Dvě akademické práce podobného zaměření publikované v posledním desetiletí pojaly téma výuky intonace a sluchové analýzy s podporou informačních technologií především z pohledu hudebně teoretického. Podněty Jiřího Hokeše a Jana Křupy k vývoji intonačních programů zůstávají cenným zdrojem přehledně zpracované hudební metodologie a návrhů na konkrétní podobu některých výukových modulů.

Při výčtu obecných požadavků kladených na intonační aplikace a následné charakteristice tří nejznámějších jsem došla k závěru, že sice existují kvalitní komerční produkty, ale při stanovení požadavku na podporu české notace a přihlédnutí k ceně se nabídka jeví jen velmi omezeně. Následně jsem se zaměřila na analýzu uživatelů a stanovila konkrétní funkční i nefunkční požadavky na vytvářenou aplikaci.

Další část práce se věnuje výběru externích knihoven a dalších komponent. Zatímco pro obecné účely jako vzhled webu nebo práce s databází jsou dostupné velmi kvalitní a snadno použitelné nástroje (nap. framework Medoo nebo Bootstrap), nalezení vhodných řešení pro notaci a syntetizaci zvuku vyžadovalo podstatně vyšší časovou dotaci. Zvláštní důraz je v práci kladen na VexFlow API, které jedním z několika málo nástrojů pro zapisování a vykreslování notace na úrovni zdrojového kódu. Jedná se o první komplexní popis tohoto API v českém jazyce.

Aplikaci jsem navrhla a implementovala pomocí programovacího jazyka PHP, přičemž staví na objektově orientovaném programování a MVC architektuře.

Stanovené cíle práce se do značné míry podařilo naplnit. Byla vytvořena webová multimediální aplikace pro nácvik intonace, která dokáže zobrazit a přehrát náhodně sestavené intonační cvičení z databáze, tudíž odpadá nutnost kontroly melodie vyučujícím nebo hudebním nástrojem. Vzniklou aplikaci jsem publikovala na adrese http://intonace.sailen.cz/, kde zůstává zájemcům veřejně přístupná.

Kromě funkce cvičení obsahuje kompletní soubor opěrných písní pro durové tóniny, a to jako notový zápis a zvukové soubory tří složek (navození tóniny, první tón a nahrávka písně) podle konvence intonačního výcviku.

Vývoj aplikace se potýkal s několika menšími problémy, z nichž ne všechny se podařilo uspokojivě vyřešit před odevzdáním této práce. Jedná se například o špatnou funkcionalitu přístupu k objektům not při SVG vykreslování ve VexFlow API, která slibovala větší interaktivitu aplikace a byla jedním z důvodů volby VexFlow. Při tvorbě práce většího rozsahu než je práce bakalářská bych zřejmě zvážila vývoj vlastního řešení vykreslování notace.

Aplikace byla úspěšně testována v několika verzích aktuálních prohlížečů, podle specifikace v příloze č. 4 by měla správně fungovat na všech moderních prohlížečích. Problémy byly zaznamenány na mobilních zařízeních v souvislosti se syntetizací zvuku, kterou z testovaných mobilních prohlížečů přehrál pouze *FlashFox*. Vykreslení SVG nepodporuje Android OS ve verzi 3.4 a nižší.

Projekt intonační aplikace získal příznivé ohlasy mezi hudebníky v mém okolí, což podněcuje snahu o jeho další rozvoj. Aplikace byla od svého počátku programována modulárně, aby bylo umožněno její další rozšiřování. V současné době je předpřipravena pro volbu rytmické obtížnosti a rozšíření od další druhy stupnic a opěrných písní.

Do budoucna bych chtěla realizovat uživatelské a učitelské účty, které by shromažďovaly data o průběhu výuky jednak pro uživatele samotné, jednak pro účely testování. Tyto účty by se následně mohly sdružovat do studijních skupin, které by poskytly novou funkcionalitu lekcí, tak jak je popsána v charakteristice zahraničních intonačních aplikací v kapitole 2.1.7. Obsahově se nabízí rozšíření o intervalový zpěv.

Výuka intonace podporovaná informačními technologiemi je v České republice nepříliš rozvinuté téma s o to větším praktickým a teoreticko-informatickým potenciálem, které by si jistě zasloužilo další rozpracování.

Souhrn

Práce se zabývá multidisciplinárním tématem výuky intonace a sluchové analýzy s podporou stále dostupnější informačních technologií. V první části stručně popisuje podobu intonačního a rytmického výcviku v České republice a obecné požadavky kladené na intonační aplikace, včetně charakteristiky tří nejznámějších softwarových produktů. Následně se zabývá tvorbou webové multimediální aplikace, která metodologicky vychází z tonální písňové metody Ladislava Daniela. Sleduje proces jejího vývoje, od analýzy uživatele a stanovení požadavků přes výběr vhodných nástrojů až po návrh a implementaci konkrétního řešení. Zvláštní důraz je zde kladen na práci s notačním API VexFlow, které v českém prostředí dosud nebylo komplexně popsáno.

Klíčová slova: Intonace, sluchová analýza, hudební výchova, webová aplikace, responzivní design, VexFlow, notační API, HTML5 audio

Summary

Subject of this thesis is education of Sight Singing and Ear Training supported by information technology which availability is growing rapidly. The first section briefly describes the convention of intonation and rhythmic training in the Czech Republic and also the general requirements for Ear Training software. There are described three well-known applications used for this purpose. Second part of thesis deals with multimedia web application methodologically based on the tonal method songs by Ladislav Daniel. The development of this application is monitored from user analysis, requirements setting and selection of appropriate tools to design and implementation of specific solution. Special importance is focused on Music Notation Rendering API VexFlow which has not been comprehensively described in Czech language yet.

Keywords: Intonation, ear training, music education, web application, responsive design, VexFlow, notation API, HTML5 audio

Terminologický slovník

Termín v angličtině	Termín v češtině	Zkratka	Význam (zdroj)
Creative Commons		CC	Sada licencí, kterými autor za určitých (v nich stanovených) podmínek veřejně zpřístupňuje své dílo a dává uživateli právo s ním nakládat. (autor)
Front-end framework			Framework, který ovlivňuje design a funkčnost té části webu, která je přístupná běžnému návštěvníkovi. (autor)
User-centred design	Design zaměřený na uživatele	UCD	Přístup k návrhu uživatelských rozhraní, který primárně zohledňuje potřeby uživatele. Dbá na to, aby vše, s čím uživatel přichází do přímého styku, bylo pro něj užitečné, snadno použitelné, pochopitelné a působilo pozitivní estetický dojmem. (Symbio, 2007 cit. podle Loukotová, 2009)
Web application	Webová aplikace		Aplikace, která je určená pro prostředí internetu a není nutné ji instalovat na uživatelské zařízení. (autor) Poskytuje svým uživatelům řešení určitého problému prostřednictvím sebe sama. (Řezáč, 2014)
Scalable Vector Graphics	Škálovatelná vektorová grafika	SVG	Vektorový grafický formát založený na technologii XML.
Wrapper	(obal)		Skript, který vytváří podmínky pro chod jiné (složitější, důležitější) funkce, je jakýmsi jejím „obalem“ zjednodušujícím přístup. (vlastní definice autora)

Seznam použité literatury a zdrojů

BOOTSTRAP. *Designed for everyone, everywhere*. [online]. 2016 [cit. 2016-03-07]. Dostupné z: http://getbootstrap.com/

DANIEL, Ladislav. *Intonace a sluchová analýza I.* 1. vyd. Olomouc: Vydavatelství Univerzity Palackého, 1994. ISBN 80-7067-351-6.

DANIEL, Ladislav. *Metodika hudební výchovy*. Vyd. 3., dopl. Ostrava: Montanex, 2010. ISBN 978-80-7225-329-6.

DANIEL, Ladislav. Výuka zpěvu z not na počítači. *e-Pedagogium.* 2003, roč. 3, č. 1 - mimořádné, s. 230-232. ISSN 1213-7758. Dostupné také z: http://epedagog.upol.cz/eped1.2003/mimo/clanek29.htm

DEVERIA, Alexis. *Can I use... Support tables for HTML5, CSS3, etc*. [online]. 2016 [cit. 2016-04-20]. Dostupné z: http://caniuse.com/

ERICSSON. *Ericsson Mobility Report. Europe*. [Online]. Sweden, Stockholm: November 2015. [Cit. 2016-02-28]. Dostupné z: http://www.ericsson.com/res/docs/2015/mobility-report/emr-nov-2015-regional-report-europe.pdf

FÍBEK, Michal. *Interaktivní e-learningová aplikace pro podporu hudební nauky na ZŠ*. Brno, 2015. Diplomová práce. Mendelova univerzita v Brně, Provozně ekonomická fakulta.

GAZDOVÁ, Blanka. Ladislav Daniel. Pokračovatel české kantorské tradice. In: *K problematice současných hudebně výchovných metod a koncepcí. Materiály z muzikologické konference k 75. narozeninám prof. dr. L. Daniela, CSc*. Olomouc: Vydavatelství Univerzity Palackého, 1998, s. 17-21. ISBN 80-7067-794-5.

HAVRÁNKOVÁ, Kateřina. Teze k didaktice HV – intonační výcvik. In: *Hudební pedagogika*. [online]. 2013-08-07 14:40 [Cit. 2015-11-10]. Dostupné z: http://hudebni-pedagogika.webnode.cz/news/intonacni-vycvik/

HOKEŠ, Jiří. *Návrh software pro výuku intonace a sluchové výchovy*. Praha, 2005. Diplomová práce. Univerzita Karlova v Praze, Pedagogická fakulta.

HORWOOD, Keith William. Musical Keyboard - JS Dynamic Audio Synth. *KeithWHor.com* [online]. © 2013 [2014] [Cit. 2016-04-15]. Dostupné z: http://www.keithwhor.com/music/

CHEPPUDIRA, Mohit Muthanna a GREG [nickname]. Re: Measures (Bars) [příspěvkz v diskuzním fóru]. In: *VexFlow: Google Groups* [online]. 18. -23. 08. 2011 [Cit. 2015-11-20]. Dostupné z: https://groups.google.com/forum/#!msg/vexflow/gQ7Zw97Zl6k/

CHEPPUDIRA, Mohit Muthanna. Re: Accessing StaveNote objects [příspěvek v diskuzním fóru]. In: *VexFlow: Google Groups* [online]. 10. 11. 2015 [Cit. 2015-11-20]. Dostupné z: https://groups.google.com/forum/#!topic/vexflow/9mv4yvEHb3A

CHEPPUDIRA, Mohit Muthanna. WexFlow. In: *GitHub* [Online]. Last updated 21. 3. 2016 [Cit. 2016-03-26]. Dostupné z: https://github.com/0xfe/vexflow

CHEPPUDIRA, Mohit Muthanna. VexFlow. Music Engraving in JavaScript and HTML5. *VexFlow.com* [Online]. ©2010-2016 [Cit. 2016-04-13]. Dostupné z: http://www.vexflow.com/docs/tutorial.html

CHEPPUDIRA, Mohit Muthanna. The VexFlow Tutorial. An n-step guide to programming with VexFlow. *VexFlow.com* [Online]. © 2010-2016 [Cit. 2016-04-13]. Dostupné z: http://www.vexflow.com/docs/tutorial.html

CHEPPUDIRA, Mohit Muthanna a RISTOW, Gregory. The VexFlow FAQ. In: *GitHub* [Online]. Last edited 4. 12. 2014 [Cit. 2016-03-26]. Dostupné z: https://github.com/0xfe/vexflow/wiki/The-VexFlow-FAQ

KŘUPA, JAN. *Nové podněty k vytváření počítačových programů pro výuku intonace a sluchové výchovy*. Praha, 2008. Diplomová práce. Pedagogická fakulta Univerzity Karlovy, Katedra hudební výchovy. Vedoucí práce Marek Valášek.

LÁTAL, Petr. *Webový portál pro výuku hudební teorie.* Zlín: 2011. Diplomová práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky.

LÁTAL, Petr, SLOVÁK, Dalibor a VAŘACHA, Pavel. Vexflow API in music learning management system. In: *Recent Researches Circuits and Systems: Proceedings of the 16th WSEAS International Conference on Circuits* [online]. WSEAS Press, 2012, s. 480-484. ISSN 1790-5117. ISBN 978-1-61804-108-1. Dostupné z: http://www.wseas.us/e-library/conferences/2012/Kos/CIRSYS/CIRSYS-83.pdf

LOUKOTOVÁ, Klára. Úvod do problematiky uživatelských rozhraní. In: ČERVENKOVÁ A. a HOŘAVA M., eds. Uživatelsky přívětivá rozhraní. Praha, Horava & Associates: 2009, s. 14 – 24. ISBN 978-80-254-5295-0.

NIELSEN ADMOSPHERE. *ATO: Vybavenost domácností moderními technologiemi roste, nejvíce posiluje smartphone*. [Online]. 2015-09-30 [Cit. 16-03-12] Dostupné z: http://www.nielsen-admosphere.cz/press/tz-ato-vybavenost-domacnosti-modernimi-technologiemi-roste-nejvice-posiluje-smartphone/

OPEN SOURCE INITIATIVE. The MIT license. *Opensource.org* [Online]. 2006 [Cit. 2016-03-20]. Dostupné z: https://opensource.org/licenses/MIT

PASTOR, Pablo. MVC for Noobs. In: *Envato* [online]. 24 Mar 2010 [cit. 2016-04-19]. Dostupné z: http://code.tutsplus.com/tutorials/mvc-for-noobs--net-10488

RISING SOFTWARE. Auralia 5. Ear training with real music. *RisingSoftware.com* [online]. [2016] [cit. 2016-04-19]. Dostupné z: https://www.risingsoftware.com/auralia/

ŘEZÁČ, Jan. *Web ostrý jako břitva. Návrh fungujícího webu pro webdesignery a zadavatele projektů*. Jihlava: Baroque partners, 2014. ISBN 978-80-87923-01-6.

THE MEDOO PROJECT. Medoo. The lightest PHP database framework to accelerate development. *Medoo.in* [online]. © 2016 [cit. 2015-11-06] Dostupné z: http://medoo.in/

VALUTIS, Osvaldas. Audio Player: Responsive and Touch-Friendly**.** *Osvaldas.info* [online] 27 DEC, 2012 [cit. 2016-02-02]. Dostupné z: http://osvaldas.info/audio-player-responsive-and-touch-friendly

WHATWG. HTML. Living Standard. *Whatwg.org* [Online]. Last updated 16. 4. 2016 [Cit. 2016-04-16]. Dostupné z: https://html.spec.whatwg.org

ZENKL, Luděk. *ABC Hudební nauky.* Vyd. 8. Praha: Editio Bärenretier, 2003. ISBN 80-86385-21-3.

Seznam obrázků, tabulek a grafů

Seznam příloh

Příloha č. 1 – VexFlow, vykreslení notové osnovy

Příklad s použitím *canvas*:

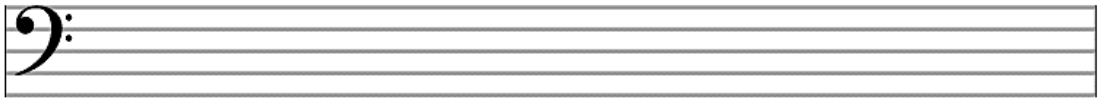
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <script src="js/vexflow-min.js"></script>
    <script src="js/jquery-2.2.3.min.js"></script>

    <title>Vykreslení prázdné notové osnovy</title>
  </head>

<body>
     <canvas id="mojeid" width=550 height=120></canvas>

        <script>
            var renderer = new Vex.Flow.Renderer(mojeid,
              Vex.Flow.Renderer.Backends.CANVAS);

            var ctx = renderer.getContext();
            var stave = new Vex.Flow.Stave(0, 0, 500);
            stave.addClef("bass").setContext(ctx).draw();       // basový klíč
        </script>
  </body>
</html>
```

Příklad s použitím *svg*:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <script src="js/vexflow-min.js"></script>
    <script src="js/jquery-2.2.3.min.js"></script>

    <title>Vykreslení prázdné notové osnovy</title>
  </head>

<body>
     <svg id="mojeid" width=550 height=120></svg>
...
```

```
...
        <script>
            var renderer = new Vex.Flow.Renderer(mojeid,
              Vex.Flow.Renderer.Backends.SVG);

            var ctx = renderer.getContext();
            var stave = new Vex.Flow.Stave(0, 0, 500);
            stave.addClef("bass").setContext(ctx).draw();       // basový klíč
        </script>
</body>
</html>
```

Výsledek:

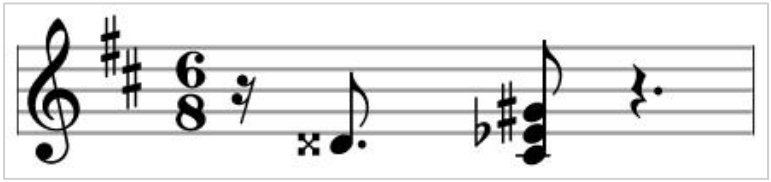


Příloha č. 2 – VexFlow, příklad zápisu s předznamenáním a posuvkami

```
<!DOCTYPE html> <html>
  <head>
    <meta charset="UTF-8">
    <script src="js/vexflow-min.js"></script>
    <script src="js/jquery-2.2.3.min.js"></script>

    <title>Vykreslení složitějšího taktu</title>
  </head>

<body>
     <svg id="mojeid" width=350 height=120></svg>

        <script>
        var renderer = new Vex.Flow.Renderer(mojeid,
              Vex.Flow.Renderer.Backends.SVG);

        var ctx = renderer.getContext();
        var stave = new Vex.Flow.Stave(0, 0, 345);
        stave.addClef("treble").setContext(ctx);          // houslový klíč
        stave.addKeySignature("D").setContext(ctx);       // předznamenání D dur
        stave.addTimeSignature("6/8").setContext(ctx);    // značení taktu 6/8
        stave.draw();

        var notes = [
        //pomlka šestnáctinová
        new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "16r" }),

        //nota disis osminová s tečkou
        new Vex.Flow.StaveNote({ keys: ["d##/4"], duration: "8d" }).
        addAccidental(0, new Vex.Flow.Accidental("##")).addDotToAll(),

        // akord c-es-gis osminový
        new Vex.Flow.StaveNote({ keys: ["c/4", "eb/4", "g#/4"], duration: "8" }).
            addAccidental(0, new Vex.Flow.Accidental("n")).
            addAccidental(1, new Vex.Flow.Accidental("b")).
            addAccidental(2, new Vex.Flow.Accidental("#")),

        // pomlka čtvrťová s tečkou
        new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "4dr" }).
            addDotToAll(),
        ];
...
```

```
...
        // Vytvoří hlas ve 4/4 taktu
        var voice = new Vex.Flow.Voice({
        num_beats: 6,
        beat_value: 8,
        resolution: Vex.Flow.RESOLUTION
        });

        // Přiřadí noty do hlasu
        voice.addTickables(notes);

        // Upraví rozložení not
        var formatter = new Vex.Flow.Formatter().
        joinVoices([voice]).format([voice], 280);

        // Vykreslí hlas
        voice.draw(ctx, stave);

         </script>
  </body>
</html>
```

Výsledek:



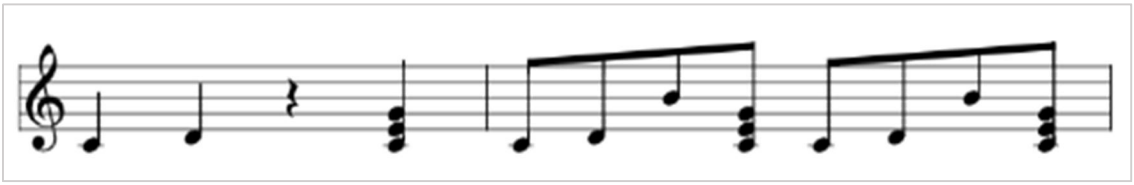
Příloha č. 3 – VexFlow, zápis více taktů pomocí juxtapozice

Zdroj: (Greg, Re: Measures (Bars), 2011), doplněno českým komentářem

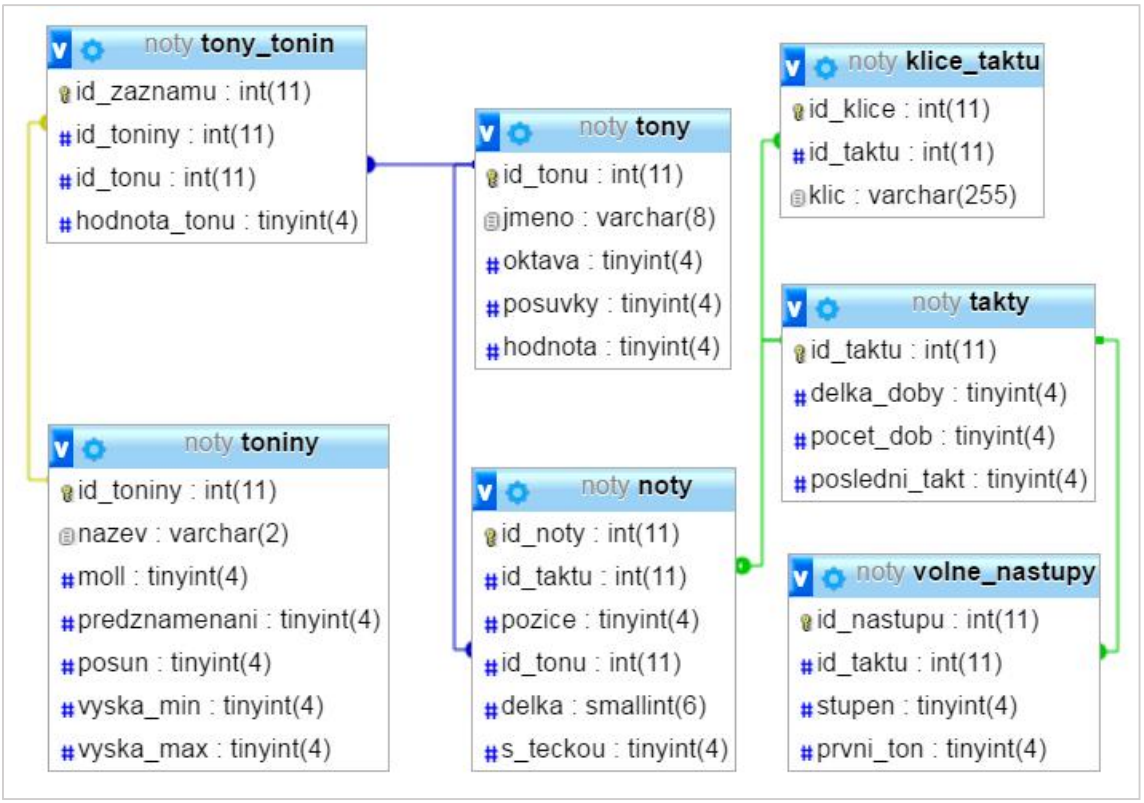
```
<!DOCTYPE html> <html>
  <head>
    <meta charset="UTF-8">
    <script src="js/vexflow-min.js"></script>
    <script src="js/jquery-2.2.3.min.js"></script>
    <title>Vykreslení složitějšího taktu</title>
  </head>
<body>
       <canvas id="score" height="300" width="800"></canvas>
<script>
$(document).ready(function() {
    var renderer = new Vex.Flow.Renderer(score,
              Vex.Flow.Renderer.Backends.CANVAS);
    var ctx = renderer.getContext();
    // takt 1
    var staveMeasure1 = new Vex.Flow.Stave(10, 0, 300);
    staveMeasure1.addClef("treble").setContext(ctx).draw();
    var notesMeasure1 = [
    new Vex.Flow.StaveNote({ keys: ["c/4"], duration: "q" }),
    new Vex.Flow.StaveNote({ keys: ["d/4"], duration: "q" }),
    new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "qr" }),
    new Vex.Flow.StaveNote({ keys: ["c/4", "e/4", "g/4"], duration: "q" })
    ];
    // Pomocná funkce k zarovnání a vykreslení hlasu
    Vex.Flow.Formatter.FormatAndDraw(ctx, staveMeasure1, notesMeasure1);
    // takt 2 - juxtapozice druhého taktu vedle prvního
    var staveMeasure2 = new Vex.Flow.Stave(staveMeasure1.
              width + staveMeasure1.x, 0, 400);
    staveMeasure2.setContext(ctx).draw();
    var notesMeasure2_part1 = [
    new Vex.Flow.StaveNote({ keys: ["c/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["d/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["c/4", "e/4", "g/4"], duration: "8" })
    ];
...
```

```
...
    var notesMeasure2_part2 = [
    new Vex.Flow.StaveNote({ keys: ["c/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["d/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["c/4", "e/4", "g/4"], duration: "8" })
    ];

    // vytvoří trámce pro osminové noty v druhém taktu
    var beam1 = new Vex.Flow.Beam(notesMeasure2_part1);
    var beam2 = new Vex.Flow.Beam(notesMeasure2_part2);

    var notesMeasure2 = notesMeasure2_part1.concat(notesMeasure2_part2);

    Vex.Flow.Formatter.FormatAndDraw(ctx, staveMeasure2, notesMeasure2);

    // vykreslí trámce
    beam1.setContext(ctx).draw();
    beam2.setContext(ctx).draw();
});

</script>
</body>
</html>
```

Výsledek:



Příloha č. 4 – ER diagram návrhu databáze



Příloha č. 5 – Podpora Web Audio API a SVG v prohlížečích k 20. 4. 2016

Zdroj: (Daveria, 2016)

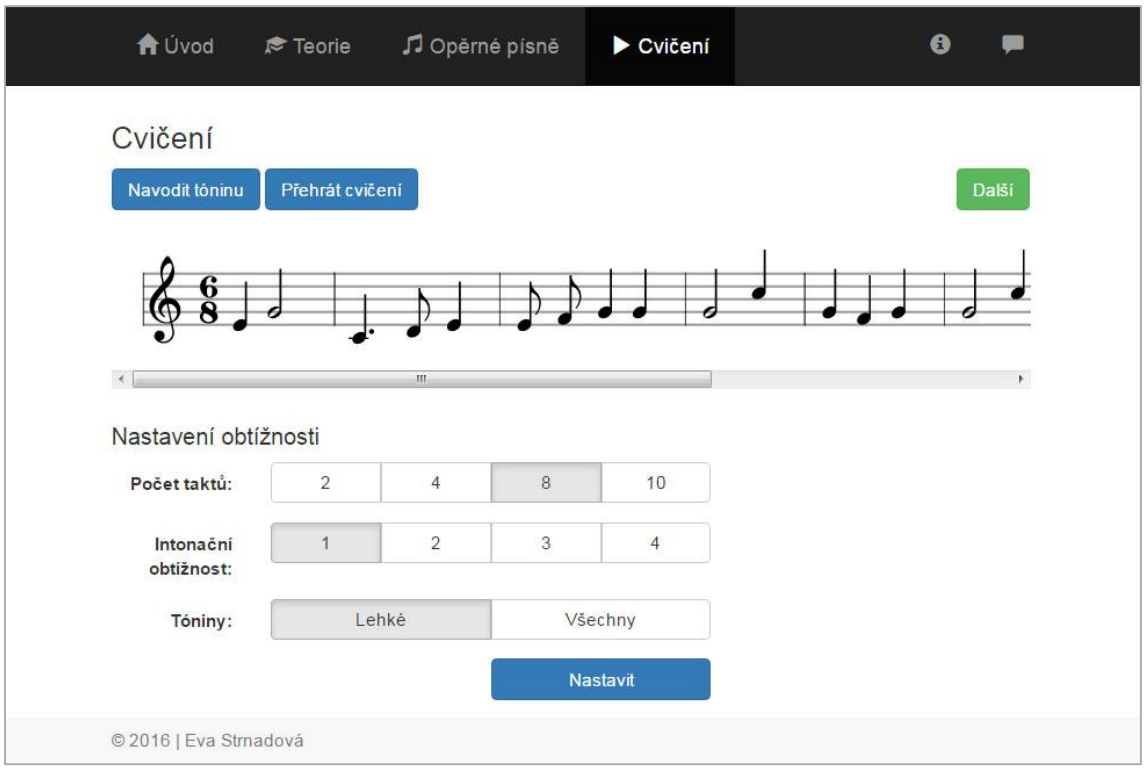
Web Audio API:

SVG (základní podpora):

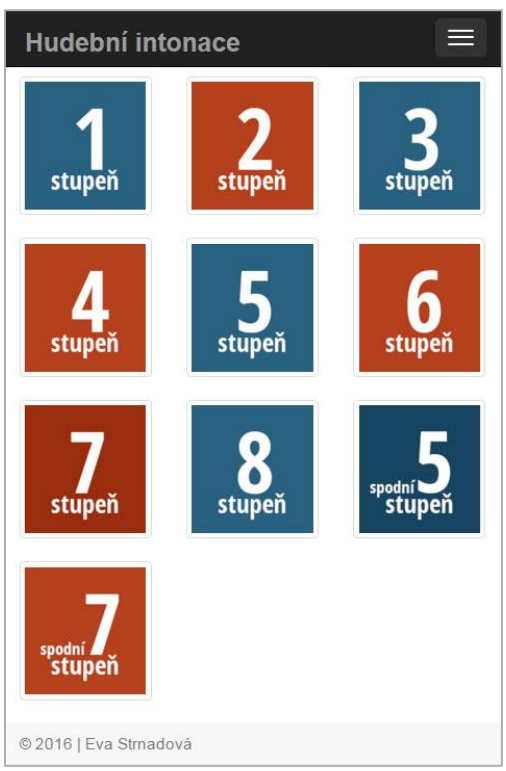
Příloha č. 6 – Uživatelské rozhraní aplikace, doplnění

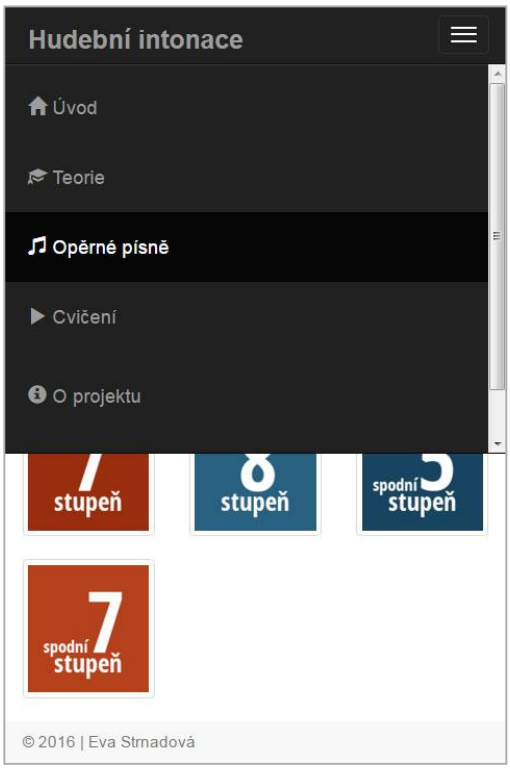
(screenshoty)

Úvodní strana, zobrazení desktop:



Menu opěrných písní a hlavní menu, zobrazení na mobilním zařízení:





Příloha č. 7 – Ukázka zdrojového kódu, třída „cviceni“

CviceniController.php

```
<?php
namespace Intonace\Cviceni;
use spectaFW as sp;
class CviceniController extends sp\Controller {

    public $model = '';
    public $pocet = 8;
    public $intonacni_obtiznost = 1;
    public $rytmicka_obtiznost = 1;
    public $tonina = 1;

    public function __construct($parametry) {
        if(isset($_POST['odeslat'])) {
            $this->pocet = $_POST['pocet_taktu'];
            $this->intonacni_obtiznost = $_POST['intonacni_obtiznost'];
            $this->rytmicka_obtiznost = $_POST['rytmicka_obtiznost'];
            $this->tonina = $_POST['tonina'];
        }

        $this->model = new Cviceni($this->pocet, $this->intonacni_obtiznost, $this-
>rytmicka_obtiznost, $this->tonina);
        $this->sablona = 'cviceni';
    }
}
```

Cviceni.php

```
<?php
namespace Intonace\Cviceni;

use Intonace\Notace;
use Intonace\Takt;
use Intonace\Rytmickaobtiznost;
use Intonace\Intonacniobtiznost;
use Intonace\Toninovaobtiznost;

class Cviceni {

    public $notace = '';
    public $intonacni_obtiznost = '';
    public $rytmicka_obtiznost = '';
    public $delka_doby = 3;
    public $pocet_dob = 4;
    public $pocet_taktu = '';
    public $toninova_obtiznost = '';
    public $obtiznost = array();

    public function __construct($pocet, $intonacni_obtiznost, $rytmicka_obtiznost,
     $toninova_obtiznost) {
        $this->pocet_taktu = $pocet;
        $this->intonacni_obtiznost = $intonacni_obtiznost;
        $this->rytmicka_obtiznost = $rytmicka_obtiznost;
        $this->toninova_obtiznost = $toninova_obtiznost;
        $this->obtiznost['rytmicka'] = new Rytmickaobtiznost\
                    Rytmickaobtiznost($this->rytmicka_obtiznost);
        $this->obtiznost['intonacni'] = new Intonacniobtiznost\
                    Intonacniobtiznost($this->intonacni_obtiznost);
        $this->obtiznost['toninova'] = new Toninovaobtiznost\
                    Toninovaobtiznost($this->toninova_obtiznost);
        $this->notace = new Notace\Notace($this->pocet_taktu, $this->obtiznost);
    }
}
```

Příloha č. 8 – Ukázka zdrojového kódu, třída „takt“

Takt.php

```
<?php
namespace Intonace\Takt;
use Intonace\Nota;
class Takt {

    public $id_taktu = '';
    public $delka_doby = '';
    public $rytmus = '';
    public $noty = array();
    public $nastupy = array();

    public function __construct($id_taktu, $delka_doby = NULL, $pocet_dob = NULL) {
        $this->id_taktu = $id_taktu;
        if($delka_doby == NULL) {
            $dotaz = TaktRepository::nactiTakt($this->id_taktu, NULL, NULL, NULL);
            $this->delka_doby = $dotaz['delka_doby'];
            $this->rytmus['pocet_dob'] = $dotaz['pocet_dob'];
        } else {
            $this->delka_doby = $delka_doby;
            $this->rytmus['pocet_dob'] = $pocet_dob;
       }
        $this->nactiNoty();
        $this->nactiNastupy();
    }
    private function nactiNoty() {
        $noty = Nota\NotaRepository::NactiNotu($this->id_taktu, NULL);
        for($i = 0; $i < count($noty); $i++) {
            $this->noty[$i] = new Nota\Nota(
                    $noty[$i]['id_noty'],
                    $noty[$i]['id_taktu'],
                    $noty[$i]['pozice'],
                    $noty[$i]['id_tonu'],
                    $noty[$i]['delka'],
                    $noty[$i]['s_teckou']);
      }
   }
    private function nactiNastupy() {
        $nastupy = TaktRepository::nactiNastupy($this->id_taktu);
        for($i = 0; $i < count($nastupy); $i++) {
            $this->nastupy[$i]['stupen'] = $nastupy[$i]['stupen'];
            $this->nastupy[$i]['prvni_ton'] = $nastupy[$i]['prvni_ton'];
    } }
...
```

```
...
public function transponuj($tonina) {
        for($i = 0; $i < count($this->noty); $i++) {
            $hodnota = $this->noty[$i]->ton->hodnota + $tonina->posun;
            for($j = 0; $j < count($tonina->tony); $j++) {
                if($tonina->tony[$j]->hodnota == $hodnota) {
                    $this->noty[$i]->zmenTon($tonina->tony[$j]->id_tonu);
                }
            }
        }
    }
    public function __toString() {
        for($i = 0; $i < count($this->noty); $i++) {
            ?>
            new Vex.Flow.StaveNote({
            keys: ["<?= $this->noty[$i]->ton->jmeno; ?>/
                        <?= $this->noty[$i]->ton->oktava; ?>"],
            duration: "<?= $this->noty[$i]->delka; ?>
            <?php
            if($this->noty[$i]->s_teckou == 1) {
                ?>
                d
                <?php
            }
            if($this->noty[$i]->ton->hodnota == 0) {
                ?>
                r
                <?php
            }
            ?>
            "})
            <?php
            if($this->noty[$i]->s_teckou == 1) {
                ?>
                .addDotToAll()
                <?php
            }
            ?>
            ,
            <?php
            return ' ';
        }
    }
}
```

TaktRespozitory.php

```
<?php
namespace Intonace\Takt;
class TaktRepository {

    static function nactiTakt($id_taktu = NULL, $delka_doby = NULL,
                        $pocet_dob = NULL, $posledni = NULL) {
        if($id_taktu != NULL) {
            $dotaz = $GLOBALS['dbMapper']->select('takty', '*',
                        array('id_taktu' => $id_taktu));
        } else {
            if($posledni != NULL) {
                $dotaz = $GLOBALS['dbMapper']->select('takty', '*', array(
                'AND' => array(
                    'delka_doby' => $delka_doby,
                    'pocet_dob' => $pocet_dob,
                    'posledni_takt' => 1)));
            } else {
                $dotaz = $GLOBALS['dbMapper']->select('takty', '*', array(
                'AND' => array(
                    'delka_doby' => $delka_doby,
                    'pocet_dob' => $pocet_dob)));
          }
       }
        if(!$dotaz) {
            die('Nepodařilo se načíst data taktu.');
        }
        return $dotaz;
   }
    static function nactiNastupy($id_taktu) {
        $dotaz = $GLOBALS['dbMapper']->select('volne_nastupy', '*',
                array('id_taktu' => $id_taktu));
        if(!$dotaz) {
            die('Nepodařilo se načíst takty.');
       }
        return $dotaz;
   }
...
```

```
...
    static function nactiRytmy() {
        $dotaz  =  $GLOBALS['dbMapper']->select('takty',  array('delka_doby',  'pocet_dob'),
array(
            'ORDER' => array('delka_doby DESC', 'pocet_dob DESC')));
        if(!$dotaz) {
            die('Nepodařilo se načíst rytmy.');
        }
        return $dotaz;
    }
}
```

Anotace

Jméno a příjmení:	Eva Strnadová
Katedra:	Katedra technické a informační výchovy
Vedoucí práce:	Mgr. Jan Kubrický, Ph.D.
Rok obhajoby:	2016

Název práce:	Multimediální aplikace pro výuku intonace
Název v angličtině:	Multimedia Application for Intonation Training
Anotace práce:	Práce se zabývá multidisciplinárním tématem výuky intonace a sluchové analýzy s podporou moderních technologií. V první části stručně popisuje podobu intonačního a rytmického výcviku v ČR a obecné požadavky kladené na intonační aplikace, včetně charakteristiky tří nejznámějších softwarových produktů. Následně se zabývá procesem vývoje webové multimediální aplikace vycházející z tonální písňové metody Ladislava Daniela se zvláštním důrazem na notační API VexFlow, které v českém prostředí dosud nebylo komplexně popsáno.
Klíčová slova:	Intonace, sluchová analýza, hudební výchova, webová aplikace, responzivní design, VexFlow, notační API, HTML5 audio
Anotace v angličtině:	Subject of this thesis is education of Sight Singing and Ear Training supported by advanced technology. The first section briefly describes the convention of intonation and rhythmic training in the Czech Republic and also the general requirements for Ear Training software. There are described three well-known applications used for this purpose. Second part of thesis deals with the development process of multimedia web application based on the tonal method songs by Ladislav Daniel. Special importance is focused on Music Notation Rendering API VexFlow which has not been comprehensively described in Czech language yet.
Klíčová slova v angličtině:	Intonation, ear training, music education, web application, responsive design, VexFlow, notation API, HTML5 audio
Přílohy vázané v práci:	1 x CD
Rozsah práce:	49 stran, 54 109 znaků
Jazyk práce:	český